phase2_tiffanywong

October 31, 2022

1 Project Phase 2

Tiffany Wong

1.0.1 Description

In this assignment, you will use the results from project phase 1 to create a dataset unique to you, which will be used for modeling in the final phase of the project.

1.0.2 Instructions

There are two parts to this assignment: evaluating annotator agreement for each data file, and assembling the files together to come up with datasets where each text has a single label that can be used for modeling. Each student is assigned a PRIMARY dataset and a SECONDARY dataset. You will need to assemble the files for the primary and secondary datasets separately, so at the end of the assignment you will have two separate files to work with. Assignments of primary and secondary datasets are in this document available in the drive link below: cs585_assignment2_dataset_assignments.csv A script to convert topic CSV files into three separate files (one for lockdowns, one for masking and distancing, one for vaccination) will be available in the drive link below with the name :- expand_topic_csv_dataset.py

[ ]:
```
# import libraries
import pandas as pd
import numpy as np
import sklearn
from sklearn.metrics import cohen_kappa_score
```

1.1 Part 1: Evaluate annotator agreement

1.1.1 Part 1

3. Each CSV file contains annotation results from 3-5 annotators, with each annotator's labels indicated in a separate column using a unique ID. For each pair of annotators, calculate the Cohen's kappa agreement score. You may use a library implementation of this function such as the one in sklearn. (For example, there will be a kappa score for (annotator_1, annotator_2), and another kappa score for (annotatot_1, annotator_3). If there are undefined or empty values in the data, treat them as a separate label (e.g., "missing")

4. Calculate a score for each individual annotator, which is the average kappa score for that annotator with all others.

```
[ ]: def all_kappa_score(dfs, stance):
    # create empty dataframe to store the kappa scores for each annotator pair
    kappa_pairs = pd.DataFrame(columns=['Dataframe', 'annotator1',␣
↪'annotator2', 'kappa score'])

    kappapair_dict = {}

    kappa_individuals = pd.DataFrame(columns=['Dataframe', 'annotator', 'kappa␣
↪score'])

    kappaindiv_dict = {}

    # iterate thru each dataframe to perform kappa cohen score for task 3 and 4␣
↪of part 1
    for idx, dataframe in enumerate(dfs):
        num_ann = len(dataframe.columns)-1

        # delete text column to access only annotation column names
        df_cols = dataframe.columns.delete(0)

        # replace all nans in dataframe
        for column in dataframe.columns:
            dataframe[column] = dataframe[column].replace(np.nan, "missing")


        # for loop for task 3 to get kappa score for each pair of annotators
        for ann_idx, anno1 in enumerate(df_cols):
            for anno2 in df_cols[ann_idx+1:]:
                # create kappa scores for each pair of annotators
                kappa_pair_score = cohen_kappa_score(dataframe[anno1],␣
↪dataframe[anno2])

                # create a dataframe type for each row with corresponding info
                kappa_row = pd.DataFrame({'Dataframe': '{stancename}_{df_idx}.
↪csv'.format(stancename=stance, df_idx=idx), 'annotator1': anno1,␣
↪'annotator2': anno2, 'kappa score': kappa_pair_score}, index=[0])
                row = [kappa_pairs, kappa_row]

                # concat new row with existing kappa dataframe
                kappa_pairs = pd.concat(row, ignore_index=True, sort=False)


        # for loop for task 4 to get individual kappa score for each annotator␣
↪in the dataframe
        for anno_idx, anno1 in enumerate(df_cols):
            kappa_sum_score = 0
            count = 0
```

```
            for anno2 in df_cols:
                if anno1 != anno2:
                    # create kappa scores for each pair of annotators
                    kappa_sum_score += cohen_kappa_score(dataframe[anno1],␣
↪dataframe[anno2])
                    count += 1

            # average the kappa score of each annotators
            kappa_individual_score = kappa_sum_score/count

            # put annotator and its score in the kappaindiv_dict
            kappaindiv_dict.update({anno1: kappa_individual_score})

            # create a dataframe type for each row with corresponding info
            kappa_row = pd.DataFrame({'Dataframe': '{stancename}_{df_idx}.csv'.
↪format(stancename=stance, df_idx=idx), 'annotator': anno1, 'kappa score':␣
↪kappa_individual_score}, index=[0])
            row = [kappa_individuals, kappa_row]

            # concat new row with existing kappa dataframe
            kappa_individuals = pd.concat(row, ignore_index=True, sort=False)

    return kappa_pairs, kappa_individuals, kappaindiv_dict
```

For Part 1: Evaluate annotator agreement task 3, I stored all of the kappa scores of the pairs of annotators in dataframes called twitter_kappa_pairs and changeorg_kappa_pairs. For task 4, I stored all the individual kappa scores of each annotator from each dataset in dataframes called twitter_kappa_individuals and changeorg_kappa_individuals.

[ ]:
```
# twitter_stance csv files
df_0 = pd.read_csv("/Users/tiffwong/Desktop/cs585/project/Datasets/
↪twitter_stance/twitter_stance_0.csv")
df_1 = pd.read_csv("/Users/tiffwong/Desktop/cs585/project/Datasets/
↪twitter_stance/twitter_stance_1.csv")
df_2 = pd.read_csv("/Users/tiffwong/Desktop/cs585/project/Datasets/
↪twitter_stance/twitter_stance_2.csv")
df_3 = pd.read_csv("/Users/tiffwong/Desktop/cs585/project/Datasets/
↪twitter_stance/twitter_stance_3.csv")
df_4 = pd.read_csv("/Users/tiffwong/Desktop/cs585/project/Datasets/
↪twitter_stance/twitter_stance_4.csv")
df_5 = pd.read_csv("/Users/tiffwong/Desktop/cs585/project/Datasets/
↪twitter_stance/twitter_stance_5.csv")

# changeorg_stance csv files
df_6 = pd.read_csv("/Users/tiffwong/Desktop/cs585/project/Datasets/
↪changeorg_stance/changeorg_stance_0.csv")
```

```
df_7 = pd.read_csv("/Users/tiffwong/Desktop/cs585/project/Datasets/
 ↪changeorg_stance/changeorg_stance_1.csv")
df_8 = pd.read_csv("/Users/tiffwong/Desktop/cs585/project/Datasets/
 ↪changeorg_stance/changeorg_stance_2.csv")
df_9 = pd.read_csv("/Users/tiffwong/Desktop/cs585/project/Datasets/
 ↪changeorg_stance/changeorg_stance_3.csv")
df_10 = pd.read_csv("/Users/tiffwong/Desktop/cs585/project/Datasets/
 ↪changeorg_stance/changeorg_stance_4.csv")

# create array of all dataframe names
dfs_twitter = [df_0, df_1, df_2, df_3, df_4, df_5]

dfs_changeorg = [df_6, df_7, df_8, df_9, df_10]

# return dataframes for both tasks's kappa scores
twitter_kappa_pairs, twitter_kappa_individuals, twitterindiv_dict =␣
 ↪all_kappa_score(dfs_twitter, 'twitter_stance')

changeorg_kappa_pairs, changeorg_kappa_individuals, changeorgindiv_dict =␣
 ↪all_kappa_score(dfs_changeorg, 'changeorg_stance')
```

Below is twitter_stance's annotators' individual kappa scores:

[ ]:
```
twitter_kappa_individuals
```

[ ]:

	Dataframe	annotator	kappa score
0	twitter_stance_0.csv	annotation_40	0.224702
1	twitter_stance_0.csv	annotation_38	0.210079
2	twitter_stance_0.csv	annotation_39	0.202508
3	twitter_stance_1.csv	annotation_48	0.297596
4	twitter_stance_1.csv	annotation_45	0.198715
5	twitter_stance_1.csv	annotation_46	0.348092
6	twitter_stance_1.csv	annotation_47	0.283584
7	twitter_stance_2.csv	annotation_64	0.244903
8	twitter_stance_2.csv	annotation_61	0.301864
9	twitter_stance_2.csv	annotation_62	0.272111
10	twitter_stance_2.csv	annotation_63	0.215491
11	twitter_stance_3.csv	annotation_56	0.298540
12	twitter_stance_3.csv	annotation_54	0.274524
13	twitter_stance_3.csv	annotation_55	0.293636
14	twitter_stance_4.csv	annotation_112	0.217123
15	twitter_stance_4.csv	annotation_113	0.257061
16	twitter_stance_4.csv	annotation_115	0.279179
17	twitter_stance_5.csv	annotation_75	0.139704
18	twitter_stance_5.csv	annotation_69	0.177966
19	twitter_stance_5.csv	annotation_70	0.159627

Below is twitter_stance's annotators' pair kappa scores:

[ ]:
```
twitter_kappa_pairs
```

[ ]:

	Dataframe	annotator1	annotator2	kappa score
0	twitter_stance_0.csv	annotation_40	annotation_38	0.232274
1	twitter_stance_0.csv	annotation_40	annotation_39	0.217131
2	twitter_stance_0.csv	annotation_38	annotation_39	0.187885
3	twitter_stance_1.csv	annotation_48	annotation_45	0.142878
4	twitter_stance_1.csv	annotation_48	annotation_46	0.438360
5	twitter_stance_1.csv	annotation_48	annotation_47	0.311550
6	twitter_stance_1.csv	annotation_45	annotation_46	0.259990
7	twitter_stance_1.csv	annotation_45	annotation_47	0.193276
8	twitter_stance_1.csv	annotation_46	annotation_47	0.345926
9	twitter_stance_2.csv	annotation_64	annotation_61	0.278484
10	twitter_stance_2.csv	annotation_64	annotation_62	0.278634
11	twitter_stance_2.csv	annotation_64	annotation_63	0.177592
12	twitter_stance_2.csv	annotation_61	annotation_62	0.347963
13	twitter_stance_2.csv	annotation_61	annotation_63	0.279146
14	twitter_stance_2.csv	annotation_62	annotation_63	0.189737
15	twitter_stance_3.csv	annotation_56	annotation_54	0.279428
16	twitter_stance_3.csv	annotation_56	annotation_55	0.317653
17	twitter_stance_3.csv	annotation_54	annotation_55	0.269620
18	twitter_stance_4.csv	annotation_112	annotation_113	0.195005
19	twitter_stance_4.csv	annotation_112	annotation_115	0.239240
20	twitter_stance_4.csv	annotation_113	annotation_115	0.319117
21	twitter_stance_5.csv	annotation_75	annotation_69	0.158043
22	twitter_stance_5.csv	annotation_75	annotation_70	0.121365
23	twitter_stance_5.csv	annotation_69	annotation_70	0.197888

Below is changeorg_stance's annotators' individual kappa scores:

[ ]:
```
changeorg_kappa_individuals
```

[ ]:

	Dataframe	annotator	kappa score
0	changeorg_stance_0.csv	annotation_106	0.185777
1	changeorg_stance_0.csv	annotation_107	0.006130
2	changeorg_stance_0.csv	annotation_108	0.184659
3	changeorg_stance_0.csv	annotation_101	0.117645
4	changeorg_stance_1.csv	annotation_73	0.180721
5	changeorg_stance_1.csv	annotation_77	0.161864
6	changeorg_stance_1.csv	annotation_78	0.172094
7	changeorg_stance_1.csv	annotation_79	0.000875
8	changeorg_stance_2.csv	annotation_84	0.471586
9	changeorg_stance_2.csv	annotation_85	0.472738
10	changeorg_stance_2.csv	annotation_86	0.493013
11	changeorg_stance_2.csv	annotation_87	0.482261
12	changeorg_stance_3.csv	annotation_97	0.318559

13	changeorg_stance_3.csv	annotation_101	0.029341
14	changeorg_stance_3.csv	annotation_102	0.280334
15	changeorg_stance_3.csv	annotation_14	0.257267
16	changeorg_stance_4.csv	annotation_92	0.016089
17	changeorg_stance_4.csv	annotation_93	0.262252
18	changeorg_stance_4.csv	annotation_94	0.287782
19	changeorg_stance_4.csv	annotation_95	0.201963

Below is twitter_stance's annotators' pair kappa scores:

[ ]:
```
changeorg_kappa_pairs
```

[ ]:

	Dataframe	annotator1	annotator2	kappa score
0	changeorg_stance_0.csv	annotation_106	annotation_107	-0.024751
1	changeorg_stance_0.csv	annotation_106	annotation_108	0.409072
2	changeorg_stance_0.csv	annotation_106	annotation_101	0.173009
3	changeorg_stance_0.csv	annotation_107	annotation_108	0.004059
4	changeorg_stance_0.csv	annotation_107	annotation_101	0.039081
5	changeorg_stance_0.csv	annotation_108	annotation_101	0.140845
6	changeorg_stance_1.csv	annotation_73	annotation_77	0.282259
7	changeorg_stance_1.csv	annotation_73	annotation_78	0.261223
8	changeorg_stance_1.csv	annotation_73	annotation_79	-0.001319
9	changeorg_stance_1.csv	annotation_77	annotation_78	0.227223
10	changeorg_stance_1.csv	annotation_77	annotation_79	-0.023891
11	changeorg_stance_1.csv	annotation_78	annotation_79	0.027835
12	changeorg_stance_2.csv	annotation_84	annotation_85	0.434757
13	changeorg_stance_2.csv	annotation_84	annotation_86	0.482607
14	changeorg_stance_2.csv	annotation_84	annotation_87	0.497396
15	changeorg_stance_2.csv	annotation_85	annotation_86	0.515252
16	changeorg_stance_2.csv	annotation_85	annotation_87	0.468205
17	changeorg_stance_2.csv	annotation_86	annotation_87	0.481181
18	changeorg_stance_3.csv	annotation_97	annotation_101	0.075536
19	changeorg_stance_3.csv	annotation_97	annotation_102	0.473006
20	changeorg_stance_3.csv	annotation_97	annotation_14	0.407137
21	changeorg_stance_3.csv	annotation_101	annotation_102	0.007909
22	changeorg_stance_3.csv	annotation_101	annotation_14	0.004579
23	changeorg_stance_3.csv	annotation_102	annotation_14	0.360087
24	changeorg_stance_4.csv	annotation_92	annotation_93	0.056454
25	changeorg_stance_4.csv	annotation_92	annotation_94	0.047657
26	changeorg_stance_4.csv	annotation_92	annotation_95	-0.055845
27	changeorg_stance_4.csv	annotation_93	annotation_94	0.442130
28	changeorg_stance_4.csv	annotation_93	annotation_95	0.288174
29	changeorg_stance_4.csv	annotation_94	annotation_95	0.373560

1.2 Part 2: Assemble datasets

1. Assign a final label to each text, according to the following logic:

1. First, eliminate any labels for annotators whose average kappa score is less than 0.2 (unreliable annotators)
2. Second, assign the final label to each text as the most frequent label among the remaining annotators
3. If there are ties (the same number of annotators for different labels), use the label with higher-reliability annotators (higher kappa scores on average)

[ ]:
```
def label_datasets(dataset_df, individual_dict, individual_df):
    labelled_dfs = pd.DataFrame()
    indices = []

    belowavg_kappa = {}

    for index, row in individual_df.iterrows():

        # look at the annotators with kappa score<0.2
        if row['kappa score'] < 0.2:
            # concat new row with existing kappa dataframe
            belowavg_kappa[row['annotator']] = row['Dataframe']


    # delete columns of annos that are unreliable
    for dataframe in dataset_df:

        # find max of the annotator's kappa scores for each dataframe, look up␣
↪each anno from the twitterindiv_dict dictionary
        all_df_anno = {}
        for annotator in dataframe.columns[1:]:
            all_df_anno.update({annotator:individual_dict[annotator]})

        maximum_kappa = max(all_df_anno.values())

        # annotator in df with greatest kappa score
        max_anno_key = list(individual_dict.keys())[list(individual_dict.
↪values()).index(maximum_kappa)]

        # create a copy of dataframe to iteratively delete bad_anno columns␣
↪from and keep original dataframes for their anno values
        new_dataframe = dataframe

        for bad_anno in belowavg_kappa.keys():
            if any(x == bad_anno for x in new_dataframe.columns):
                # drop the column of the unreliable annotator (axis=1 means␣
↪look at the columns of df)
                new_dataframe = new_dataframe.drop([bad_anno], axis=1)

        # create empty column called label in new dataframe
```

```
        new_dataframe["label"] = ""

        # create a df to evaluate mode values for
        mode_df = new_dataframe.drop(columns=["text", "label"])

        # create variable for number of annotators
        num_annos = len(mode_df.columns)

        if num_annos == 0:
            new_dataframe['label'] = dataframe[max_anno_key]

        for index, row in mode_df.mode(axis=1).iterrows():
            nan_col = mode_df.mode(axis=1)[1].isnull()
            # if mode[1] is NaN, that means theres a mode value
            if nan_col[index]:
                new_dataframe['label'][index] = row[0]
            # if mode[1] is not nan, that means there's a tie
            else:
                # check if there are 2 or 3 annotators, then mode=highest kappa␣
↪value
                if num_annos == 3 or num_annos == 2:
                    new_dataframe['label'][index] =␣
↪new_dataframe[max_anno_key][index]
                # if there are 4 annotators (the last test case), then we have␣
↪to average out each type of label value's kappa score
                else:
                    # ex: anno1, anno3 = label1 and anno2, anno4 = label2, do␣
↪the following:
                    # avg anno1, anno3 kappa scores and anno2, anno4 kappa␣
↪scores
                    # the label of the text is the label with the higher avg␣
↪kappa score
                    label1=mode_df.iloc[index].unique()[0]
                    label2=mode_df.iloc[index].unique()[1]

                    anno_label1 = mode_df.apply(lambda row: row[row == label1].
↪index, axis=1)[index]

                    avgkappa_label1 = (individual_dict[anno_label1[0]] +␣
↪individual_dict[anno_label1[1]]) / 2

                    anno_label2 = mode_df.apply(lambda row: row[row == label2].
↪index, axis=1)[index]

                    avgkappa_label2 = (individual_dict[anno_label2[0]] +␣
↪individual_dict[anno_label2[1]]) / 2
```

```
                    # evaluate if label1 or label2's avg kappa score is bigger
                    if avgkappa_label1 > avgkappa_label2:
                        new_dataframe['label'][index] = label1
                    else:
                        new_dataframe['label'][index] = label2

        # join all text and its labels for this dataset
        labelled_dfs = pd.concat([labelled_dfs, new_dataframe[['text',␣
↪'label']]], sort=False, ignore_index=True)

    return labelled_dfs
```

[ ]:
```
twitter_labels = label_datasets(dfs_twitter, twitterindiv_dict,␣
↪twitter_kappa_individuals)
changeorg_labels = label_datasets(dfs_changeorg, changeorgindiv_dict,␣
↪changeorg_kappa_individuals)
```

[ ]:
```
twitter_labels.to_csv('/Users/tiffwong/Desktop/cs585/project/
↪twitter_stance_labels.csv')
```

[ ]:
```
changeorg_labels.to_csv('/Users/tiffwong/Desktop/cs585/project/
↪changeorg_stance_labels.csv')
```