In [15]:

```
#reading the necessary libraries: #base model
import scipy.sparse as sp
from scipy.sparse import hstack
import csv
from sklearn.datasets import make_multilabel_classification
from sklearn.multioutput import MultiOutputClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import precision_score, recall_score, f1_score,classification_report
from sklearn.feature_extraction.text import TfidfVectorizer

#seeding for randamziation:
import numpy as np
np.random.seed(42)
import random
random.seed(42)
```

In [16]:

```
#reading raw data from Final_Annotationsby_Detectives_GSD.csv Golden Standards dataset:
#Feature and Label Variables:
X_txt= []
y= []

# Loading data from CSVs:
# Load the training datasets into two lists (X_txt will be a list of strings with features;
# y will list of 0's and 1's with labels or classification):
with open('./Gold Standards DataSet_D_SA.csv',encoding='iso-8859-1') as in_file:
    iCSV = csv.reader(in_file, delimiter=',')
    header=next(iCSV)
    for row in iCSV:
        X_txt.append(row[1])
        y.append([int(value) for value in row[2:6]])

#print(len(X_txt))
#print(X_txt)
#print(len(y))
#print(header[1:6],y[0])
```

In [17]:

```
#Split the data into training and test sets:
X_train,X_test,y_train,y_test=train_test_split(X_txt,y,test_size=0.2)
```

In [18]:

```
#Base Modeling with all task 1 features and task 2 lables(without lexicon features implementation):
#modeling with ngram_range=(1,1) & LogisticRegression with CountVectorizer:

#converting list to matrix:
vec=CountVectorizer(ngram_range=(1,1))
X_train_matrix =vec.fit_transform(X_train) # This should be a matrix
X_test_matrix=vec.transform(X_test)# This should be a matrix

#converting list to array:
ya_train=np.array(y_train)
ya_test=np.array(y_test)
#print(y_test.shape[1])

#initializing logisticregression:
log_reg=MultiOutputClassifier(LogisticRegression(random_state=42,solver='lbfgs', max_iter=2000))

#params with c values:
params= {"estimator__C": [0.0001, 0.001, 0.01, 0.1, 1, 10,100]}

#initialize GridSearchCV with scoring f1_macro:

init_grid_search_macro=GridSearchCV(log_reg,params,cv=5,scoring='f1_macro')

# Fit the model on X_train with scoring f1_macro:
init_grid_search_macro.fit(X_train_matrix,ya_train)

#Validation Score with scoring f1_macro:
validation_macro_score = init_grid_search_macro.score(X_test_matrix,ya_test)
validation_results_best_score=init_grid_search_macro.best_score_
# Get the score from the GridSearchCV "best score" with Macro f1:
print("Macro Validation Score F1: {:.4f}".format(validation_results_best_score))
print("Macro Test Score F1: {:.4f}".format(validation_macro_score))

#predciting on X_test data with scoring f1_macro:
logistic_X_test_prediciton_macro=init_grid_search_macro.predict(X_test_matrix)

# Calculating precision, recall and f1 Scores with average as macro parameter:
precision_macro=precision_score(ya_test,logistic_X_test_prediciton_macro,average='macro')  # Get scores using logistic_X_test_pre
recall_macro = recall_score(ya_test,logistic_X_test_prediciton_macro,average='macro')
print("Macro_Score Precision: {:.4f}".format(precision_macro))
```

```python
print("Macro_Score Recall: {:.4f}".format(recall_macro))
for i in range(4):
    #f1_macro_i = f1_score([item[i] for item in y_test], [item[i] for item in logistic_X_test_prediciton_macro])
    f1_macro_i = f1_score(ya_test[:,i],logistic_X_test_prediciton_macro[:,i])
    print(f"{header[i+2]} Macro_Score F1: {f1_macro_i:.4f}")

print()


#initialize GridSearchCV with scoring f1_micro:

init_grid_search_micro=GridSearchCV(log_reg,params,cv=5,scoring='f1_micro')

# Fit the model on X_train with scoring f1_micro:
init_grid_search_micro.fit(X_train_matrix,ya_train)

#Validation Score with scoring f1_macro:
validation_micro_score = init_grid_search_micro.score(X_test_matrix,ya_test)
validation_results_best_score_micro=init_grid_search_micro.best_score_
# Get the score from the GridSearchCV "best score" with Macro f1:
print("Micro Validation Score F1: {:.4f}".format(validation_results_best_score_micro))
print("Micro Test Score F1: {:.4f}".format(validation_micro_score))

# Get the score from the GridSearchCV "best score" with Micro f1:
#print("Micro_Score Validation F1: {:.4f}".format(validation_micro_score))

#predciting on X_test data with scoring f1_micro:
logistic_X_test_prediciton_micro=init_grid_search_micro.predict(X_test_matrix)

# Calculating precision, recall and f1 Scores with average as micro parameter:
precision_micro=precision_score(ya_test,logistic_X_test_prediciton_micro,average='micro')  # Get scores using logistic_X_test_pre
recall_micro = recall_score(ya_test,logistic_X_test_prediciton_micro,average='micro')
print("Micro_Score Precision: {:.4f}".format(precision_micro))
print("Micro_Score Recall: {:.4f}".format(recall_micro))
for i in range(4):
    #f1_micro_i = f1_score([item[i] for item in y_test], [item[i] for item in logistic_X_test_prediciton_micro])
    f1_micro_i = f1_score(ya_test[:,i],logistic_X_test_prediciton_micro[:,i])
    print(f"{header[i+2]} Micro_Score F1: {f1_micro_i:.4f}")
```

```
Macro Validation Score F1: 0.4110
Macro Test Score F1: 0.4007
Macro_Score Precision: 0.4137
Macro_Score Recall: 0.3926
Gold Standards- Technology Macro_Score F1: 0.8201
Gold Standards- Ride Share Macro_Score F1: 0.2069
Gold Standards- Food Delivery Macro_Score F1: 0.3492
Gold Standards- Online Shopping Macro_Score F1: 0.2264

Micro Validation Score F1: 0.5229
Micro Test Score F1: 0.5010
Micro_Score Precision: 0.5708
Micro_Score Recall: 0.4464
Gold Standards- Technology Micro_Score F1: 0.8319
Gold Standards- Ride Share Micro_Score F1: 0.2078
Gold Standards- Food Delivery Micro_Score F1: 0.2778
Gold Standards- Online Shopping Micro_Score F1: 0.1522
```

In [19]:

```python
#Base Modeling with all task 1 features and task 2 lables(without lexicon features implementation):
#modeling with ngram_range=(1,5) & LogisticRegression with TfidfVectorizer:

#converting list to matrix:
vec_tfid=TfidfVectorizer(ngram_range=(1,5))
X_train_matrix_tfid =vec_tfid.fit_transform(X_train) # This should be a matrix
X_test_matrix_tfid=vec_tfid.transform(X_test)# This should be a matrix


#converting list to array:
ya_train_tfid=np.array(y_train)
ya_test_tfid=np.array(y_test)
#print(y_test.shape[1])

#initializing logisticregression:
log_reg_tfid=MultiOutputClassifier(LogisticRegression(random_state=42,solver='lbfgs', max_iter=2000))

#params with c values:
params_tfid= {"estimator__C": [0.0001, 0.001, 0.01, 0.1, 1, 10,100]}

#initialize GridSearchCV with scoring f1_macro:

init_grid_search_macro_tfid=GridSearchCV(log_reg_tfid,params_tfid,cv=5,scoring='f1_macro')

# Fit the model on X_train with scoring f1_macro:
init_grid_search_macro_tfid.fit(X_train_matrix_tfid,ya_train_tfid)
```

```python
#Validation Score with scoring f1_macro:
validation_macro_score_tfid = init_grid_search_macro_tfid.score(X_test_matrix_tfid,ya_test_tfid)
validation_results_best_score_tfid=init_grid_search_macro_tfid.best_score_
# Get the score from the GridSearchCV "best score" with Macro f1:
print("Macro Validation Score F1: {:.4f}".format(validation_results_best_score_tfid))
print("Macro Test Score F1: {:.4f}".format(validation_macro_score_tfid))

#predciting on X_test data with scoring f1_macro:
logistic_X_test_prediciton_macro_tfid=init_grid_search_macro_tfid.predict(X_test_matrix_tfid)

# Calculating precision, recall and f1 Scores with average as macro parameter:
precision_macro_tfid=precision_score(ya_test_tfid,logistic_X_test_prediciton_macro_tfid,average='macro')  # Get scores using logi
recall_macro_tfid = recall_score(ya_test_tfid,logistic_X_test_prediciton_macro_tfid,average='macro')
print("Macro_Score Precision: {:.4f}".format(precision_macro_tfid))
print("Macro_Score Recall: {:.4f}".format(recall_macro_tfid))
for i_tfid in range(4):
    #f1_macro_i = f1_score([item[i] for item in y_test], [item[i] for item in logistic_X_test_prediciton_macro])
    f1_macro_i_tfid = f1_score(ya_test_tfid[:,i_tfid],logistic_X_test_prediciton_macro_tfid[:,i_tfid])
    print(f"{header[i_tfid+2]} Macro_Score F1: {f1_macro_i_tfid:.4f}")

print()


#initialize GridSearchCV with scoring f1_micro:

init_grid_search_micro_tfid=GridSearchCV(log_reg_tfid,params_tfid,cv=5,scoring='f1_micro')

# Fit the model on X_train with scoring f1_micro:
init_grid_search_micro_tfid.fit(X_train_matrix_tfid,ya_train_tfid)

#Validation Score with scoring f1_macro:
validation_micro_score_tfid = init_grid_search_micro_tfid.score(X_test_matrix_tfid,ya_test_tfid)
validation_results_best_score_micro_tfid=init_grid_search_micro_tfid.best_score_
# Get the score from the GridSearchCV "best score" with Macro f1:
print("Micro Validation Score F1: {:.4f}".format(validation_results_best_score_micro_tfid))
print("Micro Test Score F1: {:.4f}".format(validation_micro_score_tfid))

# Get the score from the GridSearchCV "best score" with Micro f1:
#print("Micro_Score Validation F1: {:.4f}".format(validation_micro_score))

#predciting on X_test data with scoring f1_micro:
logistic_X_test_prediciton_micro_tfid=init_grid_search_micro_tfid.predict(X_test_matrix_tfid)
```

```python
# Calculating precision, recall and f1 Scores with average as micro parameter:
precision_micro_tfid=precision_score(ya_test_tfid,logistic_X_test_prediciton_micro_tfid,average='micro')  # Get scores using logi
recall_micro_tfid = recall_score(ya_test_tfid,logistic_X_test_prediciton_micro_tfid,average='micro')
print("Micro_Score Precision: {:.4f}".format(precision_micro_tfid))
print("Micro_Score Recall: {:.4f}".format(recall_micro_tfid))
for i_tfid in range(4):
    #f1_micro_i = f1_score([item[i] for item in y_test], [item[i] for item in logistic_X_test_prediciton_micro])
    f1_micro_i_tfid = f1_score(ya_test_tfid[:,i_tfid],logistic_X_test_prediciton_micro_tfid[:,i_tfid])
    print(f"{header[i_tfid+2]} Micro_Score F1: {f1_micro_i_tfid:.4f}")
```

```
Macro Validation Score F1: 0.2106
Macro Test Score F1: 0.2612
Macro_Score Precision: 0.4455
Macro_Score Recall: 0.2639
Gold Standards- Technology Macro_Score F1: 0.7939
Gold Standards- Ride Share Macro_Score F1: 0.0377
Gold Standards- Food Delivery Macro_Score F1: 0.0988
Gold Standards- Online Shopping Macro_Score F1: 0.1143

Micro Validation Score F1: 0.4964
Micro Test Score F1: 0.4881
Micro_Score Precision: 0.6494
Micro_Score Recall: 0.3910
Gold Standards- Technology Micro_Score F1: 0.7839
Gold Standards- Ride Share Micro_Score F1: 0.0377
Gold Standards- Food Delivery Micro_Score F1: 0.0533
Gold Standards- Online Shopping Micro_Score F1: 0.0968
```

In [20]:

```python
#lexicon features class declaration: Tech word Count,Onlie App, Capital Words,
# Exclamation points, Positive and Negative words and their count:
class LexiconClassifier():
    def __init__(self):

        self.exclamation_points = set()
        self.online_apps_words=set()
        self.technology_app_wrd=set()
        self.food_delivery_app_wrd=set()

    def technology_wrd(self, file_path):
        with open('./technology-words.txt',encoding='iso-8859-1') as iFile:
            for row in iFile:
                word=row.strip().lower()
                self.technology_app_wrd.add(word)
                #print(word)
    def fooddelivery_app(self, file_path):
        with open('./food-delivery-words.txt',encoding='iso-8859-1') as iFile:
            for row in iFile:
                word=row.strip().lower()
                self.food_delivery_app_wrd.add(word)

    def online_apps(self, file_path):
        with open('./online-words.txt',encoding='iso-8859-1') as iFile:
            for row in iFile:
                word=row.strip().lower()
                self.online_apps_words.add(word)
               # print(row.strip())

    def load_exclamation_points(self, file_path):
        with open('./Gold Standards DataSet_D_SA.csv',encoding='iso-8859-1') as iFile:
            for row in iFile:
                word=row.strip().lower()
                self.exclamation_points.add(word)
                #print(row.strip())

    def predict_exclamation_points(self, sentence):
        """
            Returns the number of exclamation points in a string.

            Keyword arguments:
            sentence -- string (e.g., "This is great!!!")

            Returns:
            num_exclamation_points -- an integer (e.g., 3)
        """
        num_exclamation_points = 0
        for char in sentence:
            if char == '!':
                num_exclamation_points += 1
        return num_exclamation_points



    def predict_online_apps(self, sentence):

        """
        Returns True if specific online application words are found in a comment.
        Keyword arguments:
        sentence -- string (e.g., "I ordered from ubereats and doordash today.")

        Returns:
        count of online_apps_present -- 2
        """
        online_apps_count = sum(word in sentence.lower() for word in self.online_apps_words)
        return online_apps_count

    def predict_fooddelivery_apps(self, sentence):

        """
        Returns True if specific online application words are found in a comment.

        Keyword arguments:
        sentence -- string (e.g., "I ordered from ubereats and doordash today.")

        Returns:
        count of online_apps_present -- 2
        """
        fooddelivery_apps_count = sum(word in sentence.lower() for word in self.food_delivery_app_wrd)
        return fooddelivery_apps_count

    def predict_technology_app(self, sentence):

        """
        Returns True if specific online application words are found in a comment.

        Keyword arguments:
        sentence -- string (e.g., "I ordered from ubereats and doordash today.")

        Returns:
        count of online_apps_present -- 2
        """
        technology_apps_count = sum(word in sentence.lower() for word in self.technology_app_wrd)
        return technology_apps_count

# Create an instance of LexiconClassifier
#classifier_instance = LexiconClassifier()

# Call the technology_wrd method with the file path
#classifier_instance.load_capital_words('./Gold Standards DataSet_D_SA.csv')
```

In [21]:

```
#Lexicon features labels for hstack: Food delivery word count, Technology word count,Onlie App word count,
#Capital Words count, Exclamation points count,
#Positive and Negative words and their count:
# WRITE CODE HERE

# Initailze to an empty list. This will be a list of li #  Initailze to an empty list. This will be a list of lists
X_test_lexicon_features_em=[]
X_train_lexicon_features_em=[]
X_train_lexicon_features_twc=[]
X_test_lexicon_features_twc=[]
X_train_lexicon_features_ola=[]
X_test_lexicon_features_ola=[]
X_train_lexicon_features_fda=[]
X_test_lexicon_features_fda=[]
# Loop over X_txt_test
#    for each string in X_txt_test (i.e., for each item in the list), pass it to LexiconClassifiers .count_pos_words() and count_
#    append a list with the counts to X_test_lexicon_features
LexiconClassifier_v=LexiconClassifier()

#count of exclamation points:
for a in X_test:
    num_exclamation_test_count=LexiconClassifier_v .predict_exclamation_points(a)
    X_test_lexicon_features_em.append(num_exclamation_test_count)


for b in X_train:
    num_exclamation_train_count=LexiconClassifier_v .predict_exclamation_points(b)
    X_train_lexicon_features_em.append(num_exclamation_train_count)

#count of technology words count:
for u in X_test:
    num_tech_wrd_test_count=LexiconClassifier_v .predict_technology_app(u)
    X_test_lexicon_features_twc.append(num_tech_wrd_test_count)


for v in X_train:
    num_tech_wrd_train_count=LexiconClassifier_v .predict_technology_app(v)
    X_train_lexicon_features_twc.append(num_tech_wrd_train_count)


#count of online app words count:
for e in X_test:
    num_online_wrd_test_count=LexiconClassifier_v .predict_online_apps(e)
    X_test_lexicon_features_ola.append(num_online_wrd_test_count)


for f in X_train:
    num_online_wrd_train_count=LexiconClassifier_v .predict_online_apps(f)
    X_train_lexicon_features_ola.append(num_online_wrd_train_count)


#count of fooddelivery app words count:
for g in X_test:
    num_fooddel_wrd_test_count=LexiconClassifier_v .predict_fooddelivery_apps(g)
    X_test_lexicon_features_fda.append(num_fooddel_wrd_test_count)


for h in X_train:
    num_fooddel_wrd_train_count=LexiconClassifier_v .predict_fooddelivery_apps(h)
    X_train_lexicon_features_fda.append(num_fooddel_wrd_train_count)
```

In [22]:

```
#modeling with count of exclamation marks ngram_range=(1,1) & LogisticRegression with CountVectorizer:

#converting list to matrix for exclamation marks:
vec_em=CountVectorizer(ngram_range=(1,1))
X_train_matrix_em =vec_em.fit_transform(X_train) # This should be a matrix
X_test_matrix_em=vec_em.transform(X_test)# This should be a matrix

# Now we need to convert X_train_lexicon_features and X_test_lexicon_features to numpy arrays
# "hstack" X_train_lexicon_features with X_train_w_lex
# "hstack" X_test_lexicon_features with X_test_w_lex
X_train_lexicon_farray_em=np.array(X_train_lexicon_features_em).reshape(-1, 1)
X_test_lexicon_farray_em=np.array(X_test_lexicon_features_em).reshape(-1, 1)

X_train_f_em_lex=hstack([X_train_matrix_em,X_train_lexicon_farray_em])
X_test_f_em_lex=hstack([X_test_matrix_em,X_test_lexicon_farray_em])

#converting list to array:
ya_train_em=np.array(y_train)
ya_test_em=np.array(y_test)
#print(y_test.shape[1])

#initializing logisticregression:
log_reg_f_em=MultiOutputClassifier(LogisticRegression(random_state=42,solver='lbfgs', max_iter=2000))

#params with c values:
```

```python
params_em= {"estimator__C": [0.0001, 0.001, 0.01, 0.1, 1, 10,100]}

#initialize GridSearchCV with scoring f1_macro:

init_grid_search_macro_f_em=GridSearchCV(log_reg_f_em,params_em,cv=5,scoring='f1_macro')

# Fit the model on X_train with scoring f1_macro:
init_grid_search_macro_f_em.fit(X_train_f_em_lex,ya_train_em)

#Validation Score with scoring f1_macro:
validation_macro_score_f_em = init_grid_search_macro_f_em.score(X_test_f_em_lex,ya_test_em)
validation_results_best_score_f_em=init_grid_search_macro_f_em.best_score_
# Get the score from the GridSearchCV "best score" with Macro f1:
print("Macro Validation Score F1- Exclamation Feature: {:.4f}".format(validation_results_best_score_f_em))
print("Macro Test Score F1- Exclamation Feature: {:.4f}".format(validation_macro_score_f_em))

#predciting on X_test data with scoring f1_macro:
logistic_X_test_prediciton_macro_f_em=init_grid_search_macro_f_em.predict(X_test_f_em_lex)

# Calculating precision, recall and f1 Scores with average as macro parameter:
precision_macro_f_em=precision_score(ya_test_em,logistic_X_test_prediciton_macro_f_em,average='macro')  # Get scores using logist
recall_macro_f_em = recall_score(ya_test_em,logistic_X_test_prediciton_macro_f_em,average='macro')
print("Macro_Score Precision-Exclamation Feature: {:.4f}".format(precision_macro_f_em))
print("Macro_Score Recall-Exclamation Feature: {:.4f}".format(recall_macro_f_em))
for i_ma in range(4):
    #f1_macro_i = f1_score([item[i] for item in y_test], [item[i] for item in logistic_X_test_prediciton_macro])
    f1_macro_f_em_i_ma = f1_score(ya_test_em[:,i_ma],logistic_X_test_prediciton_macro_f_em[:,i_ma])
    print(f"{header[i_ma+2]} Macro_Score F1-Exclamation Feature: {f1_macro_f_em_i_ma:.4f}")

print()


#initialize GridSearchCV with scoring f1_micro:

init_grid_search_micro_f_em=GridSearchCV(log_reg_f_em,params_em,cv=5,scoring='f1_micro')

# Fit the model on X_train with scoring f1_micro:
init_grid_search_micro_f_em.fit(X_train_f_em_lex,ya_train_em)

#Validation Score with scoring f1_macro:
validation_micro_score_f_em = init_grid_search_micro_f_em.score(X_test_f_em_lex,ya_test_em)
validation_results_best_score_micro_f_em=init_grid_search_micro_f_em.best_score_
# Get the score from the GridSearchCV "best score" with Macro f1:
print("Micro Validation Score F1-Exclamation Feature: {:.4f}".format(validation_results_best_score_micro_f_em))
print("Micro Test Score F1-Exclamation Feature: {:.4f}".format(validation_micro_score_f_em))

# Get the score from the GridSearchCV "best score" with Micro f1:
#print("Micro_Score Validation F1: {:.4f}".format(validation_micro_score))

#predciting on X_test data with scoring f1_micro:
logistic_X_test_prediciton_micro_f_em=init_grid_search_micro_f_em.predict(X_test_f_em_lex)
#print(logistic_X_test_prediciton_micro_f_em)
# Calculating precision, recall and f1 Scores with average as micro parameter:
precision_micro_f_em=precision_score(ya_test_em,logistic_X_test_prediciton_micro_f_em,average='micro')  # Get scores using logist
recall_micro_f_em = recall_score(ya_test_em,logistic_X_test_prediciton_micro_f_em,average='micro')
print("Micro_Score Precision-Exclamation Feature: {:.4f}".format(precision_micro_f_em))
print("Micro_Score Recall-Exclamation Feature: {:.4f}".format(recall_micro_f_em))
for i_mi in range(4):
    #f1_micro_i = f1_score([item[i] for item in y_test], [item[i] for item in logistic_X_test_prediciton_micro])
    f1_micro_f_em_i_mi = f1_score(ya_test_em[:,i_mi],logistic_X_test_prediciton_micro_f_em[:,i_mi])
    print(f"{header[i_mi+2]} Micro_Score F1-Exclamation Feature: {f1_micro_f_em_i_mi:.4f}")
```

```
Macro Validation Score F1- Exclamation Feature: 0.4115
Macro Test Score F1- Exclamation Feature: 0.3990
Macro_Score Precision-Exclamation Feature: 0.4113
Macro_Score Recall-Exclamation Feature: 0.3926
Gold Standards- Technology Macro_Score F1-Exclamation Feature: 0.8099
Gold Standards- Ride Share Macro_Score F1-Exclamation Feature: 0.2069
Gold Standards- Food Delivery Macro_Score F1-Exclamation Feature: 0.3548
Gold Standards- Online Shopping Macro_Score F1-Exclamation Feature: 0.2243

Micro Validation Score F1-Exclamation Feature: 0.5208
Micro Test Score F1-Exclamation Feature: 0.4982
Micro_Score Precision-Exclamation Feature: 0.5292
Micro_Score Recall-Exclamation Feature: 0.4706
Gold Standards- Technology Micro_Score F1-Exclamation Feature: 0.8151
Gold Standards- Ride Share Micro_Score F1-Exclamation Feature: 0.1882
Gold Standards- Food Delivery Micro_Score F1-Exclamation Feature: 0.3471
Gold Standards- Online Shopping Micro_Score F1-Exclamation Feature: 0.1961
```

In [23]:

```
#modeling with count of technology word count ngram_range=(1,2) & LogisticRegression with CountVectorizer:

#converting list to matrix for technology word count:
vec_twc=CountVectorizer(ngram_range=(1,2))
X_train_matrix_twc =vec_twc.fit_transform(X_train) # This should be a matrix
X_test_matrix_twc=vec_twc.transform(X_test)# This should be a matrix

# Now we need to convert X_train_lexicon_features and X_test_lexicon_features to numpy arrays
```

```python
# "hstack" X_train_lexicon_features with X_train_w_lex
# "hstack" X_test_lexicon_features with X_test_w_lex
X_train_lexicon_farray_twc=np.array(X_train_lexicon_features_twc).reshape(-1, 1)
X_test_lexicon_farray_twc=np.array(X_test_lexicon_features_twc).reshape(-1, 1)

X_train_f_twc_lex=hstack([X_train_matrix_twc,X_train_lexicon_farray_twc])
X_test_f_twc_lex=hstack([X_test_matrix_twc,X_test_lexicon_farray_twc])

#converting list to array:
ya_train_twc=np.array(y_train)
ya_test_twc=np.array(y_test)
#print(y_test.shape[1])

#initializing logisticregression:
log_reg_f_twc=MultiOutputClassifier(LogisticRegression(random_state=42,solver='lbfgs', max_iter=2000))

#params with c values:
params_twc= {"estimator__C": [0.0001, 0.001, 0.01, 0.1, 1, 10,100]}

#initialize GridSearchCV with scoring f1_macro:

init_grid_search_macro_f_twc=GridSearchCV(log_reg_f_twc,params_twc,cv=5,scoring='f1_macro')

# Fit the model on X_train with scoring f1_macro:
init_grid_search_macro_f_twc.fit(X_train_f_twc_lex,ya_train_twc)

#Validation Score with scoring f1_macro:
validation_macro_score_f_twc = init_grid_search_macro_f_twc.score(X_test_f_twc_lex,ya_test_twc)
validation_results_best_score_f_twc=init_grid_search_macro_f_twc.best_score_
# Get the score from the GridSearchCV "best score" with Macro f1:
print("Macro Validation Score F1- Tech word count Feature: {:.4f}".format(validation_results_best_score_f_twc))
print("Macro Test Score F1- Tech word count Feature: {:.4f}".format(validation_macro_score_f_twc))

#predciting on X_test data with scoring f1_macro:
logistic_X_test_prediciton_macro_f_twc=init_grid_search_macro_f_twc.predict(X_test_f_twc_lex)

# Calculating precision, recall and f1 Scores with average as macro parameter:
precision_macro_f_twc=precision_score(ya_test_twc,logistic_X_test_prediciton_macro_f_twc,average='macro')  # Get scores using log
recall_macro_f_twc = recall_score(ya_test_twc,logistic_X_test_prediciton_macro_f_twc,average='macro')
print("Macro_Score Precision-Tech word count Feature: {:.4f}".format(precision_macro_f_twc))
print("Macro_Score Recall-Tech word count Feature: {:.4f}".format(recall_macro_f_twc))
for i_twc_mac in range(4):
    #f1_macro_i = f1_score([item[i] for item in y_test], [item[i] for item in logistic_X_test_prediciton_macro])
    f1_macro_f_twc_i_twc = f1_score(ya_test_twc[:,i_twc_mac],logistic_X_test_prediciton_macro_f_twc[:,i_twc_mac])
    print(f"{header[i_twc_mac+2]} Macro_Score F1-Tech word count Feature: {f1_macro_f_twc_i_twc:.4f}")

print()


#initialize GridSearchCV with scoring f1_micro:

init_grid_search_micro_f_twc=GridSearchCV(log_reg_f_twc,params_twc,cv=5,scoring='f1_micro')

# Fit the model on X_train with scoring f1_micro:
init_grid_search_micro_f_twc.fit(X_train_f_twc_lex,ya_train_twc)

#Validation Score with scoring f1_macro:
validation_micro_score_f_twc = init_grid_search_micro_f_twc.score(X_test_f_twc_lex,ya_test_twc)
validation_results_best_score_micro_f_twc=init_grid_search_micro_f_twc.best_score_
# Get the score from the GridSearchCV "best score" with Macro f1:
print("Micro Validation Score F1-Tech word count Feature: {:.4f}".format(validation_results_best_score_micro_f_twc))
print("Micro Test Score F1-Tech word count Feature: {:.4f}".format(validation_micro_score_f_twc))

# Get the score from the GridSearchCV "best score" with Micro f1:
#print("Micro_Score Validation F1: {:.4f}".format(validation_micro_score))

#predciting on X_test data with scoring f1_micro:
logistic_X_test_prediciton_micro_f_twc=init_grid_search_micro_f_twc.predict(X_test_f_twc_lex)

# Calculating precision, recall and f1 Scores with average as micro parameter:
precision_micro_f_twc=precision_score(ya_test_twc,logistic_X_test_prediciton_micro_f_twc,average='micro')  # Get scores using log
recall_micro_f_twc = recall_score(ya_test_twc,logistic_X_test_prediciton_micro_f_twc,average='micro')
print("Micro_Score Precision-Tech word count Feature: {:.4f}".format(precision_micro_f_twc))
print("Micro_Score Recall-Tech word count Feature: {:.4f}".format(recall_micro_f_twc))
for i_twc_mic in range(4):
    #f1_micro_i = f1_score([item[i] for item in y_test], [item[i] for item in logistic_X_test_prediciton_micro])
    f1_micro_f_twc_i_twc_mic = f1_score(ya_test_twc[:,i_twc_mic],logistic_X_test_prediciton_micro_f_twc[:,i_twc_mic])
    print(f"{header[i_twc_mic+2]} Micro_Score F1-Tech word count Feature: {f1_micro_f_twc_i_twc_mic:.4f}")
```

```
Macro Validation Score F1- Tech word count Feature: 0.3796
Macro Test Score F1- Tech word count Feature: 0.3429
Macro_Score Precision-Tech word count Feature: 0.3956
Macro_Score Recall-Tech word count Feature: 0.3240
Gold Standards- Technology Macro_Score F1-Tech word count Feature: 0.8230
Gold Standards- Ride Share Macro_Score F1-Tech word count Feature: 0.1176
Gold Standards- Food Delivery Macro_Score F1-Tech word count Feature: 0.2752
Gold Standards- Online Shopping Macro_Score F1-Tech word count Feature: 0.1556

Micro Validation Score F1-Tech word count Feature: 0.5212
Micro Test Score F1-Tech word count Feature: 0.5010
Micro_Score Precision-Tech word count Feature: 0.6089
Micro_Score Recall-Tech word count Feature: 0.4256
Gold Standards- Technology Micro_Score F1-Tech word count Feature: 0.8197
Gold Standards- Ride Share Micro_Score F1-Tech word count Feature: 0.1818
Gold Standards- Food Delivery Micro_Score F1-Tech word count Feature: 0.2353
Gold Standards- Online Shopping Micro_Score F1-Tech word count Feature: 0.1266
```

In [24]:

```
#modeling with count of online app word count ngram_range=(2,3) & LogisticRegression with CountVectorizer:

#converting list to matrix for technology word count:
vec_olp=CountVectorizer(ngram_range=(2,3))
X_train_matrix_olp =vec_olp.fit_transform(X_train) # This should be a matrix
X_test_matrix_olp=vec_olp.transform(X_test)# This should be a matrix

# Now we need to convert X_train_lexicon_features and X_test_lexicon_features to numpy arrays
# "hstack" X_train_lexicon_features with X_train_w_lex
# "hstack" X_test_lexicon_features with X_test_w_lex
X_train_lexicon_farray_olp=np.array(X_train_lexicon_features_ola).reshape(-1, 1)
X_test_lexicon_farray_olp=np.array(X_test_lexicon_features_ola).reshape(-1, 1)

X_train_f_olp_lex=hstack([X_train_matrix_olp,X_train_lexicon_farray_olp])
X_test_f_olp_lex=hstack([X_test_matrix_olp,X_test_lexicon_farray_olp])

#converting list to array:
ya_train_olp=np.array(y_train)
ya_test_olp=np.array(y_test)
#print(y_test.shape[1])

#initializing logisticregression:
log_reg_f_olp=MultiOutputClassifier(LogisticRegression(random_state=42,solver='lbfgs', max_iter=2000))

#params with c values:
params_olp= {"estimator__C": [0.0001, 0.001, 0.01, 0.1, 1, 10,100]}
```

```
#initialize GridSearchCV with scoring f1_macro:

init_grid_search_macro_f_olp=GridSearchCV(log_reg_f_olp,params_olp,cv=5,scoring='f1_macro')

# Fit the model on X_train with scoring f1_macro:
init_grid_search_macro_f_olp.fit(X_train_f_olp_lex,ya_train_olp)

#Validation Score with scoring f1_macro:
validation_macro_score_f_olp = init_grid_search_macro_f_olp.score(X_test_f_olp_lex,ya_test_olp)
validation_results_best_score_f_olp=init_grid_search_macro_f_olp.best_score_
# Get the score from the GridSearchCV "best score" with Macro f1:
print("Macro Validation Score F1- online app word count Feature: {:.4f}".format(validation_results_best_score_f_olp))
print("Macro Test Score F1- online app word count Feature: {:.4f}".format(validation_macro_score_f_olp))

#predciting on X_test data with scoring f1_macro:
logistic_X_test_prediciton_macro_f_olp=init_grid_search_macro_f_olp.predict(X_test_f_olp_lex)

# Calculating precision, recall and f1 Scores with average as macro parameter:
precision_macro_f_olp=precision_score(ya_test_olp,logistic_X_test_prediciton_macro_f_olp,average='macro')  # Get scores using log
recall_macro_f_olp = recall_score(ya_test_olp,logistic_X_test_prediciton_macro_f_olp,average='macro')
print("Macro_Score Precision-online app word count Feature: {:.4f}".format(precision_macro_f_olp))
print("Macro_Score Recall-online app word count Feature: {:.4f}".format(recall_macro_f_olp))
for i_olp_mac in range(4):
    #f1_macro_i = f1_score([item[i] for item in y_test], [item[i] for item in logistic_X_test_prediciton_macro])
    f1_macro_f_olp_i_olp = f1_score(ya_test_olp[:,i_olp_mac],logistic_X_test_prediciton_macro_f_olp[:,i_olp_mac])
    print(f"{header[i_olp_mac+2]} Macro_Score F1-online app word count Feature: {f1_macro_f_olp_i_olp:.4f}")

print()


#initialize GridSearchCV with scoring f1_micro:

init_grid_search_micro_f_olp=GridSearchCV(log_reg_f_olp,params_olp,cv=5,scoring='f1_micro')

# Fit the model on X_train with scoring f1_micro:
init_grid_search_micro_f_olp.fit(X_train_f_olp_lex,ya_train_olp)

#Validation Score with scoring f1_macro:
validation_micro_score_f_olp = init_grid_search_micro_f_olp.score(X_test_f_olp_lex,ya_test_olp)
validation_results_best_score_micro_f_olp=init_grid_search_micro_f_olp.best_score_
# Get the score from the GridSearchCV "best score" with Macro f1:
print("Micro Validation Score F1-online app word count Feature: {:.4f}".format(validation_results_best_score_micro_f_olp))
print("Micro Test Score F1-online app word count Feature: {:.4f}".format(validation_micro_score_f_olp))
```

```

# Get the score from the GridSearchCV "best score" with Micro f1:
#print("Micro_Score Validation F1: {:.4f}".format(validation_micro_score))

#predciting on X_test data with scoring f1_micro:
logistic_X_test_prediciton_micro_f_olp=init_grid_search_micro_f_olp.predict(X_test_f_olp_lex)

# Calculating precision, recall and f1 Scores with average as micro parameter:
precision_micro_f_olp=precision_score(ya_test_olp,logistic_X_test_prediciton_micro_f_olp,average='micro')  # Get scores using log
recall_micro_f_olp = recall_score(ya_test_olp,logistic_X_test_prediciton_micro_f_olp,average='micro')
print("Micro_Score Precision-online app word count Feature: {:.4f}".format(precision_micro_f_olp))
print("Micro_Score Recall-online app word count Feature: {:.4f}".format(recall_micro_f_olp))
for i_olp_mic in range(4):
    #f1_micro_i = f1_score([item[i] for item in y_test], [item[i] for item in logistic_X_test_prediciton_micro])
    f1_micro_f_olp_i_olp_mic = f1_score(ya_test_olp[:,i_olp_mic],logistic_X_test_prediciton_micro_f_olp[:,i_olp_mic])
    print(f"{header[i_olp_mic+2]} Micro_Score F1-online app word count Feature: {f1_micro_f_olp_i_olp_mic:.4f}")
```

```
Macro Validation Score F1- online app word count Feature: 0.2504
Macro Test Score F1- online app word count Feature: 0.2398
Macro_Score Precision-online app word count Feature: 0.4013
Macro_Score Recall-online app word count Feature: 0.2634
Gold Standards- Technology Macro_Score F1-online app word count Feature: 0.7535
Gold Standards- Ride Share Macro_Score F1-online app word count Feature: 0.0741
Gold Standards- Food Delivery Macro_Score F1-online app word count Feature: 0.0460
Gold Standards- Online Shopping Macro_Score F1-online app word count Feature: 0.0857

Micro Validation Score F1-online app word count Feature: 0.4967
Micro Test Score F1-online app word count Feature: 0.4807
Micro_Score Precision-online app word count Feature: 0.5842
Micro_Score Recall-online app word count Feature: 0.4083
Gold Standards- Technology Micro_Score F1-online app word count Feature: 0.7331
Gold Standards- Ride Share Micro_Score F1-online app word count Feature: 0.0385
Gold Standards- Food Delivery Micro_Score F1-online app word count Feature: 0.0286
Gold Standards- Online Shopping Micro_Score F1-online app word count Feature: 0.0690
```

In [25]:

```
#modeling with count of food delivery app word count ngram_range=(1,4) &
#LogisticRegression with CountVectorizer:

#converting list to matrix for technology word count:
vec_fda=CountVectorizer(ngram_range=(1,4))
X_train_matrix_fda =vec_fda.fit_transform(X_train) # This should be a matrix
X_test_matrix_fda=vec_fda.transform(X_test)# This should be a matrix

# Now we need to convert X_train_lexicon_features and X_test_lexicon_features to numpy arrays
# "hstack" X_train_lexicon_features with X_train_w_lex
# "hstack" X_test_lexicon_features with X_test_w_lex
X_train_lexicon_farray_fda=np.array(X_train_lexicon_features_fda).reshape(-1, 1)
X_test_lexicon_farray_fda=np.array(X_test_lexicon_features_fda).reshape(-1, 1)

X_train_f_fda_lex=hstack([X_train_matrix_fda,X_train_lexicon_farray_fda])
X_test_f_fda_lex=hstack([X_test_matrix_fda,X_test_lexicon_farray_fda])

#converting list to array:
ya_train_fda=np.array(y_train)
ya_test_fda=np.array(y_test)
#print(y_test.shape[1])

#initializing logisticregression:
log_reg_f_fda=MultiOutputClassifier(LogisticRegression(random_state=42,solver='lbfgs', max_iter=2000))

#params with c values:
params_fda= {"estimator__C": [0.0001, 0.001, 0.01, 0.1, 1, 10,100]}

#initialize GridSearchCV with scoring f1_macro:

init_grid_search_macro_f_fda=GridSearchCV(log_reg_f_fda,params_fda,cv=5,scoring='f1_macro')

# Fit the model on X_train with scoring f1_macro:
init_grid_search_macro_f_fda.fit(X_train_f_fda_lex,ya_train_fda)

#Validation Score with scoring f1_macro:
validation_macro_score_f_fda = init_grid_search_macro_f_fda.score(X_test_f_fda_lex,ya_test_fda)
validation_results_best_score_f_fda=init_grid_search_macro_f_fda.best_score_
# Get the score from the GridSearchCV "best score" with Macro f1:
print("Macro Validation Score F1- Food delivery app word count Feature: {:.4f}".format(validation_results_best_score_f_fda))
print("Macro Test Score F1- Food delivery app word count Feature: {:.4f}".format(validation_macro_score_f_fda))

#predciting on X_test data with scoring f1_macro:
logistic_X_test_prediciton_macro_f_fda=init_grid_search_macro_f_fda.predict(X_test_f_fda_lex)

# Calculating precision, recall and f1 Scores with average as macro parameter:
precision_macro_f_fda=precision_score(ya_test_fda,logistic_X_test_prediciton_macro_f_fda,average='macro')  # Get scores using log
recall_macro_f_fda = recall_score(ya_test_fda,logistic_X_test_prediciton_macro_f_fda,average='macro')
print("Macro_Score Precision-Food delivery app word count Feature: {:.4f}".format(precision_macro_f_fda))
print("Macro_Score Recall-Food delivery app word count Feature: {:.4f}".format(recall_macro_f_fda))
#for i_fda_mac in range(4):
    #f1_macro_i = f1_score([item[i] for item in y_test], [item[i] for item in logistic_X_test_prediciton_macro])
i_fda_mac=2
```

```python
f1_macro_f_fda_i_fda = f1_score(ya_test_fda[:,i_fda_mac],logistic_X_test_prediciton_macro_f_fda[:,i_fda_mac])
print(f"{header[i_fda_mac+2]} Macro_Score F1-Food delivery app word count Feature: {f1_macro_f_fda_i_fda:.4f}")

print()



#initialize GridSearchCV with scoring f1_micro:

init_grid_search_micro_f_fda=GridSearchCV(log_reg_f_fda,params_fda,cv=5,scoring='f1_micro')

# Fit the model on X_train with scoring f1_micro:
init_grid_search_micro_f_fda.fit(X_train_f_fda_lex,ya_train_fda)

#Validation Score with scoring f1_macro:
validation_micro_score_f_fda = init_grid_search_micro_f_fda.score(X_test_f_fda_lex,ya_test_fda)
validation_results_best_score_micro_f_fda=init_grid_search_micro_f_fda.best_score_
# Get the score from the GridSearchCV "best score" with Macro f1:
print("Micro Validation Score F1-Food delivery app word count Feature: {:.4f}".format(validation_results_best_score_micro_f_fda))
print("Micro Test Score F1-Food delivery app word count Feature: {:.4f}".format(validation_micro_score_f_fda))

# Get the score from the GridSearchCV "best score" with Micro f1:
#print("Micro_Score Validation F1: {:.4f}".format(validation_micro_score))

#predciting on X_test data with scoring f1_micro:
logistic_X_test_prediciton_micro_f_fda=init_grid_search_micro_f_fda.predict(X_test_f_fda_lex)

# Calculating precision, recall and f1 Scores with average as micro parameter:
precision_micro_f_fda=precision_score(ya_test_fda,logistic_X_test_prediciton_micro_f_fda,average='micro')  # Get scores using log
recall_micro_f_fda = recall_score(ya_test_fda,logistic_X_test_prediciton_micro_f_fda,average='micro')
print("Micro_Score Precision-Food delivery app word count Feature: {:.4f}".format(precision_micro_f_fda))
print("Micro_Score Recall-Food delivery app word count Feature: {:.4f}".format(recall_micro_f_fda))
#for i_fda_mic in range(3):
    #f1_micro_i = f1_score([item[i] for item in y_test], [item[i] for item in logistic_X_test_prediciton_micro])
i_fda_mic=2
f1_micro_f_fda_i_fda_mic = f1_score(ya_test_fda[:,i_fda_mic],logistic_X_test_prediciton_micro_f_fda[:,i_fda_mic])
print(f"{header[i_fda_mic+2]} Micro_Score F1-Food delivery app word count Feature: {f1_micro_f_fda_i_fda_mic:.4f}")
```

```
Macro Validation Score F1- Food delivery app word count Feature: 0.3429
Macro Test Score F1- Food delivery app word count Feature: 0.3216
Macro_Score Precision-Food delivery app word count Feature: 0.4368
Macro_Score Recall-Food delivery app word count Feature: 0.2927
Gold Standards- Food Delivery Macro_Score F1-Food delivery app word count Feature: 0.1429

Micro Validation Score F1-Food delivery app word count Feature: 0.5131
Micro Test Score F1-Food delivery app word count Feature: 0.4820
Micro_Score Precision-Food delivery app word count Feature: 0.6196
Micro_Score Recall-Food delivery app word count Feature: 0.3945
Gold Standards- Food Delivery Micro_Score F1-Food delivery app word count Feature: 0.1443
```

In [13]:

```python
#modeling with count of food delivery app word count ngram_range=(1,4) &
#LogisticRegression with TfidfVectorizer:

#converting list to matrix for technology word count:
vec_fda_tfid=TfidfVectorizer(ngram_range=(1,4))
X_train_matrix_fda_tfid =vec_fda_tfid.fit_transform(X_train) # This should be a matrix
X_test_matrix_fda_tfid=vec_fda_tfid.transform(X_test)# This should be a matrix

# Now we need to convert X_train_lexicon_features and X_test_lexicon_features to numpy arrays
# "hstack" X_train_lexicon_features with X_train_w_lex
# "hstack" X_test_lexicon_features with X_test_w_lex
X_train_lexicon_farray_fda_tfid=np.array(X_train_lexicon_features_fda).reshape(-1, 1)
X_test_lexicon_farray_fda_tfid=np.array(X_test_lexicon_features_fda).reshape(-1, 1)

X_train_f_fda_lex_tfid=hstack([X_train_matrix_fda_tfid,X_train_lexicon_farray_fda_tfid])
X_test_f_fda_lex_tfid=hstack([X_test_matrix_fda_tfid,X_test_lexicon_farray_fda_tfid])

#converting list to array:
ya_train_fda_tfid=np.array(y_train)
ya_test_fda_tfid=np.array(y_test)
#print(y_test.shape[1])

#initializing logisticregression:
log_reg_f_fda_tfid=MultiOutputClassifier(LogisticRegression(random_state=42,solver='lbfgs', max_iter=2000))

#params with c values:
params_fda_tfid= {"estimator__C": [0.0001, 0.001, 0.01, 0.1, 1, 10,100]}

#initialize GridSearchCV with scoring f1_macro:

init_grid_search_macro_f_fda_tfid=GridSearchCV(log_reg_f_fda_tfid,params_fda_tfid,cv=5,scoring='f1_macro')
```

```
# Fit the model on X_train with scoring f1_macro:
init_grid_search_macro_f_fda_tfid.fit(X_train_f_fda_lex_tfid,ya_train_fda_tfid)

#Validation Score with scoring f1_macro:
validation_macro_score_f_fda_tfid = init_grid_search_macro_f_fda_tfid.score(X_test_f_fda_lex_tfid,ya_test_fda_tfid)
validation_results_best_score_f_fda_tfid=init_grid_search_macro_f_fda_tfid.best_score_
# Get the score from the GridSearchCV "best score" with Macro f1:
print("Macro Validation Score F1- Food delivery app word count Feature: {:.4f}".format(validation_results_best_score_f_fda_tfid))
print("Macro Test Score F1- Food delivery app word count Feature: {:.4f}".format(validation_macro_score_f_fda_tfid))

#predciting on X_test data with scoring f1_macro:
logistic_X_test_prediciton_macro_f_fda_tfid=init_grid_search_macro_f_fda_tfid.predict(X_test_f_fda_lex_tfid)

# Calculating precision, recall and f1 Scores with average as macro parameter:
precision_macro_f_fda_tfid=precision_score(ya_test_fda_tfid,logistic_X_test_prediciton_macro_f_fda_tfid,average='macro')  # Get s
recall_macro_f_fda_tfid = recall_score(ya_test_fda_tfid,logistic_X_test_prediciton_macro_f_fda_tfid,average='macro')
print("Macro_Score Precision-Food delivery app word count Feature: {:.4f}".format(precision_macro_f_fda_tfid))
print("Macro_Score Recall-Food delivery app word count Feature: {:.4f}".format(recall_macro_f_fda_tfid))
#for i_fda_mac in range(4):
    #f1_macro_i = f1_score([item[i] for item in y_test], [item[i] for item in logistic_X_test_prediciton_macro])
i_fda_mac_tfid=2
f1_macro_f_fda_i_fda_tfid = f1_score(ya_test_fda_tfid[:,i_fda_mac_tfid],logistic_X_test_prediciton_macro_f_fda_tfid[:,i_fda_mac_t
print(f"{header[i_fda_mac_tfid+2]} Macro_Score F1-Food delivery app word count Feature: {f1_macro_f_fda_i_fda_tfid:.4f}")

print()


#initialize GridSearchCV with scoring f1_micro:

init_grid_search_micro_f_fda_tfid=GridSearchCV(log_reg_f_fda_tfid,params_fda_tfid,cv=5,scoring='f1_micro')

# Fit the model on X_train with scoring f1_micro:
init_grid_search_micro_f_fda_tfid.fit(X_train_f_fda_lex_tfid,ya_train_fda_tfid)

#Validation Score with scoring f1_macro:
validation_micro_score_f_fda_tfid = init_grid_search_micro_f_fda_tfid.score(X_test_f_fda_lex_tfid,ya_test_fda_tfid)
validation_results_best_score_micro_f_fda_tfid=init_grid_search_micro_f_fda_tfid.best_score_
# Get the score from the GridSearchCV "best score" with Macro f1:
print("Micro Validation Score F1-Food delivery app word count Feature: {:.4f}".format(validation_results_best_score_micro_f_fda_t
print("Micro Test Score F1-Food delivery app word count Feature: {:.4f}".format(validation_micro_score_f_fda_tfid))

# Get the score from the GridSearchCV "best score" with Micro f1:
#print("Micro_Score Validation F1: {:.4f}".format(validation_micro_score))
```

```
#predciting on X_test data with scoring f1_micro:
logistic_X_test_prediciton_micro_f_fda_tfid=init_grid_search_micro_f_fda_tfid.predict(X_test_f_fda_lex_tfid)

# Calculating precision, recall and f1 Scores with average as micro parameter:
precision_micro_f_fda_tfid=precision_score(ya_test_fda_tfid,logistic_X_test_prediciton_micro_f_fda_tfid,average='micro')  # Get s
recall_micro_f_fda_tfid = recall_score(ya_test_fda_tfid,logistic_X_test_prediciton_micro_f_fda_tfid,average='micro')
print("Micro_Score Precision-Food delivery app word count Feature: {:.4f}".format(precision_micro_f_fda_tfid))
print("Micro_Score Recall-Food delivery app word count Feature: {:.4f}".format(recall_micro_f_fda_tfid))
#for i_fda_mic in range(3):
    #f1_micro_i = f1_score([item[i] for item in y_test], [item[i] for item in logistic_X_test_prediciton_micro])
i_fda_mic_tfid=2
f1_micro_f_fda_i_fda_mic_tfid = f1_score(ya_test_fda_tfid[:,i_fda_mic_tfid],logistic_X_test_prediciton_micro_f_fda_tfid[:,i_fda_m
print(f"{header[i_fda_mic_tfid+2]} Micro_Score F1-Food delivery app word count Feature: {f1_micro_f_fda_i_fda_mic_tfid:.4f}")
```

```
Macro Validation Score F1- Food delivery app word count Feature: 0.2109
Macro Test Score F1- Food delivery app word count Feature: 0.2498
Macro_Score Precision-Food delivery app word count Feature: 0.4631
Macro_Score Recall-Food delivery app word count Feature: 0.2622
Gold Standards- Food Delivery Macro_Score F1-Food delivery app word count Feature: 0.0779

Micro Validation Score F1-Food delivery app word count Feature: 0.4976
Micro Test Score F1-Food delivery app word count Feature: 0.4903
Micro_Score Precision-Food delivery app word count Feature: 0.6477
Micro_Score Recall-Food delivery app word count Feature: 0.3945
Gold Standards- Food Delivery Micro_Score F1-Food delivery app word count Feature: 0.0779
```

In [32]:

```
# Manual analysis of the predictions to verify:
num_tweets = 0
for comment,logistic_Xtest_prediciton_micro_f_em,logistic_Xtest_prediciton_macro_f_em,logistic_Xtest_prediciton_macro_f_twc,logis
    print("Tweet: {}".format(comment))
    print(header[2:6])
    print("Ground-Truth Class: {}".format(y))
    print("Lexicon Exclamation Model Prediction(micro f1): {}".format(logistic_Xtest_prediciton_micro_f_em))
    print("Lexicon Exclamation Model Prediction(macro f1): {}".format(logistic_Xtest_prediciton_macro_f_em))
    print("Lexicon Technology Word Count Model Prediction(macro f1): {}".format(logistic_Xtest_prediciton_macro_f_twc))
    print("Lexicon Technology Word Count Model Prediction(micro f1): {}".format(logistic_Xtest_prediciton_micro_f_twc))
    print("Lexicon Online App Word Count Model Prediction(micro f1): {}".format(logistic_Xtest_prediciton_micro_f_olp))
    print("Lexicon Online App Word Count Model Prediction(macro f1): {}".format(logistic_Xtest_prediciton_macro_f_olp))
    print("Lexicon Food Delivery App Word Count Model Prediction(micro f1): {}".format(logistic_Xtest_prediciton_micro_f_fda))
    print("Lexicon Food Delivery App Word Count Model Prediction(macro f1): {}".format(logistic_Xtest_prediciton_macro_f_fda))
    print()

    num_tweets += 1
    if num_tweets == 10:
        break
```

```
Tweet: I think arts, culture, and science should be subsidized as it often appears that the free market doesn't grow those areas
effectively and authentically.  But considering how cheap modern technology has made some things, such as making movies and tv s
hows, it may not be as necessary.
['Gold Standards- Technology', 'Gold Standards- Ride Share', 'Gold Standards- Food Delivery', 'Gold Standards- Online Shopping']
Ground-Truth Class: [0, 0, 0, 0]
Lexicon Exclamation Model Prediction(micro f1): [1 0 1 0]
Lexicon Exclamation Model Prediction(macro f1): [1 0 1 0]
Lexicon Technology Word Count Model Prediction(macro f1): [1 0 0 1]
Lexicon Technology Word Count Model Prediction(micro f1): [1 0 0 0]
Lexicon Online App Word Count Model Prediction(micro f1): [1 0 0 0]
Lexicon Online App Word Count Model Prediction(macro f1): [1 0 0 0]
Lexicon Food Delivery App Word Count Model Prediction(micro f1): [1 0 0 0]
Lexicon Food Delivery App Word Count Model Prediction(macro f1): [1 0 0 0]

Tweet: If you wish to stick with iPhone, the iPhone 6S is still currently supported with iOS 15. You can get one used for about
$75 on eBay and maybe even cheaper on a local marketplace app
['Gold Standards- Technology', 'Gold Standards- Ride Share', 'Gold Standards- Food Delivery', 'Gold Standards- Online Shopping']
Ground-Truth Class: [1, 0, 0, 1]
Lexicon Exclamation Model Prediction(micro f1): [0 1 1 0]
Lexicon Exclamation Model Prediction(macro f1): [0 1 1 0]
Lexicon Technology Word Count Model Prediction(macro f1): [0 0 1 0]
Lexicon Technology Word Count Model Prediction(micro f1): [1 0 1 0]
Lexicon Online App Word Count Model Prediction(micro f1): [1 0 0 0]
Lexicon Online App Word Count Model Prediction(macro f1): [1 0 0 0]
Lexicon Food Delivery App Word Count Model Prediction(micro f1): [1 0 1 0]
Lexicon Food Delivery App Word Count Model Prediction(macro f1): [0 1 1 0]

Tweet: I know, its just harder than I thought living with my parents driving me up a wall.  When I was working; I was fine becau
se I would literally be working 10-12 hours a day, get home, make something to eat, watch some Netflix or play video games, go t
o sleep, and repeat.  Not a whole lot of interactions except to organize a family movie night at a theater with my brother as we
ll and not having it be so late for us to still go to bed and wake up the next morning at 5 or so...

Now I'm taking some online classes in computer science and business so I can hopefully learn and be able land something to be ab
le to still work from home and making a pay check next time something like this happens.  I work for a wholesale business that d
elivers liquor to bars and restaurants - but when they shut down, business went down as well and we were out of a job for a wee
k.  Luckily, our boss was also opening up 2 new stores before all this and is giving us some work to do to both help them out as
well as us whenever this clears out and we can hit the ground running better prepared.  They're still going to pay us our full a
mount that we were getting before, just a different kind of work.  Still work.
['Gold Standards- Technology', 'Gold Standards- Ride Share', 'Gold Standards- Food Delivery', 'Gold Standards- Online Shopping']
Ground-Truth Class: [0, 0, 1, 0]
Lexicon Exclamation Model Prediction(micro f1): [0 0 0 0]
Lexicon Exclamation Model Prediction(macro f1): [0 0 0 0]
Lexicon Technology Word Count Model Prediction(macro f1): [0 0 0 0]
Lexicon Technology Word Count Model Prediction(micro f1): [0 0 0 0]
Lexicon Online App Word Count Model Prediction(micro f1): [1 0 0 0]
Lexicon Online App Word Count Model Prediction(macro f1): [1 0 0 0]
Lexicon Food Delivery App Word Count Model Prediction(micro f1): [0 0 0 0]
Lexicon Food Delivery App Word Count Model Prediction(macro f1): [0 0 0 0]

Tweet: I'm really stuck as to where to begin, but I'll stay brief:

You officially rationalized intrusion. Even if it is as you say it is (it isn't), it doesn't make up for the principal of the fa
ct. True, google's algorithms can figure out a lot. Wtf do you think the government is using?! Bringing up how bad other governm
ents are doesn't make what the government is doing "okay". If anything, the government knows that keeping it's people comfortabl
e allows them far more freedom to fuck over other countries than if they wanted to treat us like they were N.Korea.

People will fight for their fake "freedom" (usa) a lot harder than for an imposing government (n.korea) (staying within the conf
ines of your examples).

This all goes much deeper, and we're hardly doing any side of the argument justice, here. If it was as you say it is, the bevy o
f guilty companies being mined wouldn't be as high as it is.
['Gold Standards- Technology', 'Gold Standards- Ride Share', 'Gold Standards- Food Delivery', 'Gold Standards- Online Shopping']
Ground-Truth Class: [1, 0, 0, 0]
Lexicon Exclamation Model Prediction(micro f1): [1 0 0 1]
Lexicon Exclamation Model Prediction(macro f1): [1 0 0 1]
Lexicon Technology Word Count Model Prediction(macro f1): [1 0 0 0]
Lexicon Technology Word Count Model Prediction(micro f1): [1 0 0 0]
Lexicon Online App Word Count Model Prediction(micro f1): [1 0 0 0]
Lexicon Online App Word Count Model Prediction(macro f1): [1 0 0 0]
Lexicon Food Delivery App Word Count Model Prediction(micro f1): [1 0 0 0]
Lexicon Food Delivery App Word Count Model Prediction(macro f1): [1 0 0 0]

Tweet: USAA is a great choice. I worked there for two years. They don�t have the most modern or fast paced environment (banking
regulations = slow) . Their technology stack is actually better than most other banks and the people who work there are pretty s
mart cookies. One of my former coworker could easily have worked at Google. The pay is subpar but for a first job it�s 11/10 whe
n it comes to benefits and culture. Your manager cares about you and your coworkers always want you to grow.
['Gold Standards- Technology', 'Gold Standards- Ride Share', 'Gold Standards- Food Delivery', 'Gold Standards- Online Shopping']
Ground-Truth Class: [1, 0, 0, 0]
Lexicon Exclamation Model Prediction(micro f1): [1 0 1 1]
Lexicon Exclamation Model Prediction(macro f1): [1 0 1 1]
Lexicon Technology Word Count Model Prediction(macro f1): [1 0 1 1]
Lexicon Technology Word Count Model Prediction(micro f1): [1 0 1 1]
Lexicon Online App Word Count Model Prediction(micro f1): [1 0 0 0]
Lexicon Online App Word Count Model Prediction(macro f1): [1 0 1 1]
Lexicon Food Delivery App Word Count Model Prediction(micro f1): [1 0 1 1]
Lexicon Food Delivery App Word Count Model Prediction(macro f1): [1 0 1 1]

Tweet: USAA is a great choice. I worked there for two years. They don�t have the most modern or fast paced environment (banking
regulations = slow) . Their technology stack is actually better than most other banks and the people who work there are pretty s
mart cookies. One of my former coworker could easily have worked at Google. The pay is subpar but for a first job it�s 11/10 whe
n it comes to benefits and culture. Your manager cares about you and your coworkers always want you to grow.
['Gold Standards- Technology', 'Gold Standards- Ride Share', 'Gold Standards- Food Delivery', 'Gold Standards- Online Shopping']
Ground-Truth Class: [1, 0, 0, 0]
Lexicon Exclamation Model Prediction(micro f1): [1 0 0 1]
Lexicon Exclamation Model Prediction(macro f1): [1 0 0 1]
Lexicon Technology Word Count Model Prediction(macro f1): [1 0 0 1]
Lexicon Technology Word Count Model Prediction(micro f1): [1 0 0 0]
Lexicon Online App Word Count Model Prediction(micro f1): [1 0 0 0]
Lexicon Online App Word Count Model Prediction(macro f1): [1 0 0 1]
Lexicon Food Delivery App Word Count Model Prediction(micro f1): [1 0 0 1]
Lexicon Food Delivery App Word Count Model Prediction(macro f1): [1 0 0 1]

Tweet: Apparently Reddit is Facebook now...
['Gold Standards- Technology', 'Gold Standards- Ride Share', 'Gold Standards- Food Delivery', 'Gold Standards- Online Shopping']
Ground-Truth Class: [1, 1, 0, 0]
Lexicon Exclamation Model Prediction(micro f1): [0 0 0 0]
Lexicon Exclamation Model Prediction(macro f1): [0 0 0 0]
Lexicon Technology Word Count Model Prediction(macro f1): [0 0 0 0]
Lexicon Technology Word Count Model Prediction(micro f1): [0 0 0 0]
Lexicon Online App Word Count Model Prediction(micro f1): [0 0 0 0]
Lexicon Online App Word Count Model Prediction(macro f1): [0 0 0 0]
Lexicon Food Delivery App Word Count Model Prediction(micro f1): [0 0 0 0]
Lexicon Food Delivery App Word Count Model Prediction(macro f1): [0 1 0 0]

Tweet: Y'all are aware that he's a Senator who spends 90% of his time away from Texas (in DC) anyway, right?  Was he supposed to
fly down here and freeze with us in unity?  Come down here and take up even more gas, water and power resources that we don't ha
ve?  lol, c'mon now.   It would be one thing if there was actually something of substance for him to do down here. But there rea
lly isn't.   All he can do is make phone calls and send emails.  And he can do that from anywhere.
['Gold Standards- Technology', 'Gold Standards- Ride Share', 'Gold Standards- Food Delivery', 'Gold Standards- Online Shopping']
Ground-Truth Class: [0, 1, 0, 0]
Lexicon Exclamation Model Prediction(micro f1): [1 0 0 1]
Lexicon Exclamation Model Prediction(macro f1): [1 0 0 1]
Lexicon Technology Word Count Model Prediction(macro f1): [0 0 0 0]
Lexicon Technology Word Count Model Prediction(micro f1): [0 0 0 0]
Lexicon Online App Word Count Model Prediction(micro f1): [1 0 0 0]
Lexicon Online App Word Count Model Prediction(macro f1): [1 0 0 0]
Lexicon Food Delivery App Word Count Model Prediction(micro f1): [0 0 0 0]
Lexicon Food Delivery App Word Count Model Prediction(macro f1): [0 0 0 0]

Tweet: They aren't officially open until Jan, but these folks are great.  Highly recommend!

https://www.facebook.com/hqmobilesalon
```

```
['Gold Standards- Technology', 'Gold Standards- Ride Share', 'Gold Standards- Food Delivery', 'Gold Standards- Online Shopping']
Ground-Truth Class: [1, 1, 1, 0]
Lexicon Exclamation Model Prediction(micro f1): [1 0 0 0]
Lexicon Exclamation Model Prediction(macro f1): [1 0 0 0]
Lexicon Technology Word Count Model Prediction(macro f1): [1 0 0 0]
Lexicon Technology Word Count Model Prediction(micro f1): [1 0 0 0]
Lexicon Online App Word Count Model Prediction(micro f1): [1 0 0 0]
Lexicon Online App Word Count Model Prediction(macro f1): [1 0 0 0]
Lexicon Food Delivery App Word Count Model Prediction(micro f1): [1 0 0 0]
Lexicon Food Delivery App Word Count Model Prediction(macro f1): [1 0 0 0]

Tweet: This is patently false, Harris and Dallas counties have much higher rates.

Source: https://txdshs.maps.arcgis.com/apps/opsdashboard/index.html#/ed483ecd702b4298ab01e8b9cafc8b83

And you dare smear our beautiful rivers, you self-loathing miser?
['Gold Standards- Technology', 'Gold Standards- Ride Share', 'Gold Standards- Food Delivery', 'Gold Standards- Online Shopping']
Ground-Truth Class: [0, 1, 1, 0]
Lexicon Exclamation Model Prediction(micro f1): [0 0 0 0]
Lexicon Exclamation Model Prediction(macro f1): [0 0 0 0]
Lexicon Technology Word Count Model Prediction(macro f1): [0 0 0 0]
Lexicon Technology Word Count Model Prediction(micro f1): [0 0 0 0]
Lexicon Online App Word Count Model Prediction(micro f1): [1 0 0 0]
Lexicon Online App Word Count Model Prediction(macro f1): [1 0 0 0]
Lexicon Food Delivery App Word Count Model Prediction(micro f1): [0 0 0 0]
Lexicon Food Delivery App Word Count Model Prediction(macro f1): [0 0 0 0]
```