term_term_pr_distr_and_inference

November 13, 2020

autoreload modules and utilities

[1]:
```
%load_ext autoreload
%autoreload 2
```

import all neceesary libraries/packages

[2]:
```
import joblib

import numpy as np
import pandas as pd

from tqdm.notebook import tqdm
import matplotlib.pyplot as plt

from sklearn.datasets import fetch_20newsgroups
from sklearn.model_selection import StratifiedShuffleSplit

from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer

from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import f1_score as calculate_f1_score
from sklearn.model_selection import train_test_split, StratifiedKFold
```

Utility functions

[116]:
```
## utilities
# from utils import clean_text

import string

from sklearn.base import TransformerMixin

import nltk
from nltk import word_tokenize
from nltk.stem import WordNetLemmatizer
```

```
nltk.download('stopwords')
nltk.download('wordnet')

wordnet_lemmatizer = WordNetLemmatizer()

def clean_text(text: str, lemmatizer = None) -> str:
    # removes upper cases
    text = text.lower().strip()

    # removes punctuation
    for char in string.punctuation:
        text = text.replace(char, "")

    #lematize the words and join back into string text
    if lemmatizer is not None:
        text = " ".join([lemmatizer(word) for word in word_tokenize(text)])

    return text

def calculate_sparsity(matrix):
    non_zero = np.count_nonzero(matrix)
    total_val = np.product(matrix.shape)
    sparsity = (total_val - non_zero) / total_val
    return sparsity

def data_isvalid(text, analyser, min_character_size, max_character_size):
    return min_character_size <= len(analyser(text)) <= max_character_size

def get_pipeline(count_vectorizer, classifier, lemmatizer, t2pi_transformer,␣
↪tfidf_transformer):
    models = [
        ('clean_text', CleanTextTransformer(lemmatizer)),
        ('vectorizer', count_vectorizer)
    ]

    if tfidf_transformer is not None:
        models.append(('tfidf_transformer', tfidf_transformer))

    models.append(
        ('dense', DenseTransformer(count_vectorizer=count_vectorizer))
    )

    if t2pi_transformer is not None:
        models.append(('t2pi_transformer', t2pi_transformer))

    models.append(('classifier', classifier))
```

```
    return Pipeline(models)

def get_model(classifier, tfidf=False, use_t2pi=False, lemmatizer=None,␣
↪return_t2pi=False, stop_words="english"):
    count_vectorizer = CountVectorizer(stop_words=stop_words)
    tfidf_transformer = TfidfTransformer() if tfidf else None
    t2pi_transformer = T2PITransformer() if use_t2pi else None

    if return_t2pi:
        return get_pipeline(count_vectorizer, classifier, lemmatizer,␣
↪t2pi_transformer, tfidf_transformer), t2pi_transformer

    # normal model
    return get_pipeline(count_vectorizer, classifier, lemmatizer,␣
↪t2pi_transformer, tfidf_transformer)

class CleanTextTransformer(TransformerMixin):
    def __init__(self, lemmatizer):
        self._lemmatizer = lemmatizer

    def fit(self, X, y=None, **fit_params):
        return self

    def transform(self, X, y=None, **fit_params):
        return np.vectorize(lambda x: clean_text(x, self._lemmatizer))(X)

    def __str__(self):
        return "CleanTextTransformer()"

    def __repr__(self):
        return self.__str__()

class DenseTransformer(TransformerMixin):
    def __init__(self, count_vectorizer):
        self.count_vectorizer = count_vectorizer

    def fit(self, X, y=None, **fit_params):
        return self

    def transform(self, X, y=None, **fit_params):
        return pd.DataFrame(data=X.todense(), columns=self.count_vectorizer.
↪get_feature_names())

    def __str__(self):
        return "DenseTransformer()"

    def __repr__(self):
```

```
        return self.__str__()

class T2PITransformer(TransformerMixin):
    @staticmethod
    def _sum_weight(x, pbar, word_word_pr_distr_prime):
        pbar.update(1)
        return word_word_pr_distr_prime.apply(lambda y: x*y, axis=0).sum(0)

    def fit(self, X, y=None, **fit_params):
        word_doc_count = X.sum(0)
        word_word_pr_distr = pd.DataFrame(data=0.0, columns=X.columns, index=X.
 ↪columns)

        print("creating term-term co-occurence pr matrix")
        for column in tqdm(X.columns):
            pxy = X[X[column] > 0].sum(0) / word_doc_count[column]
            word_word_pr_distr[column] = pxy * (word_doc_count[column] /␣
 ↪word_doc_count)

        # scale to integers
        min_value = word_word_pr_distr[word_word_pr_distr > 0].min().min()
        self.word_word_pr_distr = word_word_pr_distr / min_value

        return self

    def transform(self, X, y=None, **fit_params):
        print("transforming ...")

        # new_sparsity after transform
        sparsity_before = calculate_sparsity(X)

        with tqdm(total=X.shape[0]) as pbar:
            X = X.apply(self._sum_weight, axis=1, args=(pbar, self.
 ↪word_word_pr_distr))

        # new_sparsity after transform
        sparsity_after = calculate_sparsity(X)

        print("sparsity(X):")
        print(f"=> before {sparsity_before:.4f}")
        print(f"=>  after {sparsity_after:.4f}")
        print()

        return X

    def __str__(self):
        return "T2PITransformer()"
```

```
    def __repr__(self):
        return self.__str__()
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\christian\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\christian\AppData\Roaming\nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
```

1 Load Data

[139]:
```
# total number of samples needed
randomize = False

# retrieve dataset
categories = ['rec.autos', 'talk.politics.mideast', 'alt.atheism', 'sci.space']

all_docs = fetch_20newsgroups(subset='train', shuffle=randomize,␣
↪remove=('headers', 'footers', 'quotes'), categories=categories)
categories = all_docs.target_names
```

[140]:
```
print(all_docs.data[0])
```

```
I think that domestication will change behavior to a large degree.
Domesticated animals exhibit behaviors not found in the wild.  I
don't think that they can be viewed as good representatives of the
wild animal kingdom, since they have been bred for thousands of years
to produce certain behaviors, etc.
```

1.0.1 Create Dataframe

[141]:
```
data = pd.DataFrame(
    data={
        "text":all_docs.data,
        "label":all_docs.target
    }
)

data.head()
```

[141]:

	text	label
0	\n\nI think that domestication will change beh…	0
1	\nI don't like this comment about "Typical" th…	3

```
2  \n<apparently you're not a woman - my husband …      1
3  While not exactly a service incident, I had a …      1
4  \n\nI think I can. Largely as a result of effo…      2
```

1.0.2 Label Frequency

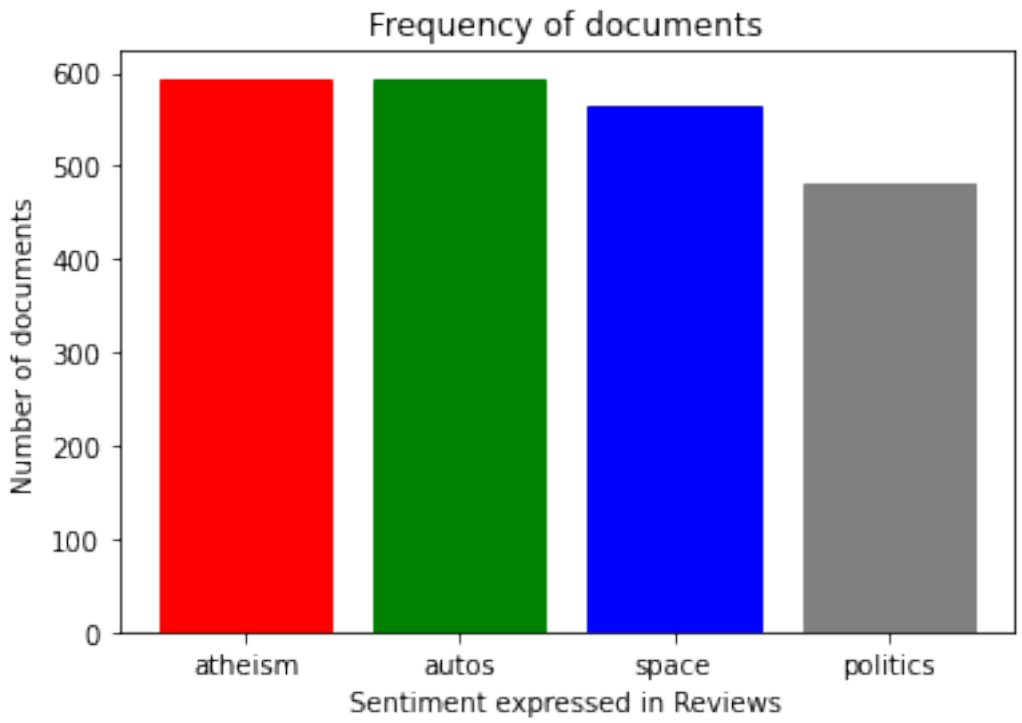
[142]:
```
print(data["label"].value_counts())
print()

barlist = plt.bar(categories, data["label"].value_counts())

plt.title("Frequency of documents")
plt.xticks(categories, list(map(lambda x: x.split(".")[1], categories)))
plt.ylabel('Number of documents')
plt.xlabel('Sentiment expressed in Reviews')

barlist[0].set_color('red')
barlist[1].set_color('green')
barlist[2].set_color('blue')
barlist[3].set_color('grey')
plt.show()
```

```
1    594
2    593
3    564
0    480
Name: label, dtype: int64
```



The Dataset labels needs to be balanced

2 Select Valid Data

[143]:
```
max_size_per_class = 100

# remove long text
indices = data["text"].apply(data_isvalid, args=(lambda x: clean_text(x,␣
 ↪wordnet_lemmatizer.lemmatize), 256, 512))
data = data[indices]

# make classes balanced
class_indices = []

for index in range(4):
    class_indices.append(np.where((data["label"] == index))[0])

size_per_class = min(max_size_per_class, min(map(len, class_indices)))
indices = np.concatenate([class_ids[:size_per_class] for class_ids in␣
 ↪class_indices])

data = data.iloc[indices]
```

```
data.head()
```

[143]:

	text	label
0	\n\nI think that domestication will change beh…	0
30	\n[rest deleted…]\n\nYou were a liberal arts…	0
36	\nWorse? Maybe not, but it is definately a vi…	0
63	\nCould you expand on your definition of knowi…	0
65	\nLooking at historical evidence such 'perfect…	0

[144]:
```
print(data.iloc[0]["text"])
```

```
I think that domestication will change behavior to a large degree.
Domesticated animals exhibit behaviors not found in the wild.  I
don't think that they can be viewed as good representatives of the
wild animal kingdom, since they have been bred for thousands of years
to produce certain behaviors, etc.
```

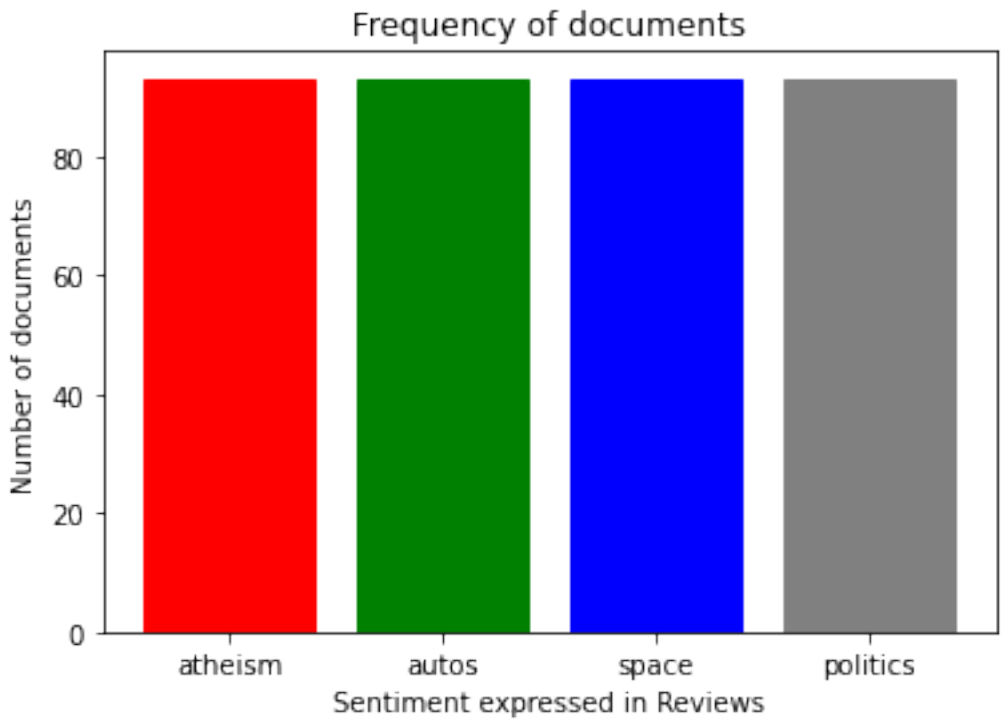
[145]:
```
print(data["label"].value_counts())
print()

barlist = plt.bar(categories, data["label"].value_counts())

plt.title("Frequency of documents")
plt.xticks(categories, list(map(lambda x: x.split(".")[1], categories)))
plt.ylabel('Number of documents')
plt.xlabel('Sentiment expressed in Reviews')

barlist[0].set_color('red')
barlist[1].set_color('green')
barlist[2].set_color('blue')
barlist[3].set_color('grey')
plt.show()
```

```
3    93
2    93
1    93
0    93
Name: label, dtype: int64
```



2.0.1 initialize input and output

[146]:
```
X = data["text"]
y = data['label']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
↪random_state=42)
```

2.0.2 initialize recursive word infer model

[147]:
```
def get_classifier():
    return MultinomialNB()
```

[148]:
```
# initialize model
t2pi_model, t2pi_transformer = get_model(use_t2pi=True, return_t2pi=True,␣
↪classifier=get_classifier())

# fit model
t2pi_model.fit(X_train, y_train)
```

```
creating term-term co-occurence pr matrix
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4204.0),␣
↪HTML(value='')))
```

```
transforming …
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=279.0),␣
 ↪HTML(value='')))
```

```
sparsity(X):
=> before 0.9931
=>  after 0.5207
```

[148]:
```
Pipeline(steps=[('clean_text', CleanTextTransformer()),
                ('vectorizer', CountVectorizer(stop_words='english')),
                ('dense', DenseTransformer()),
                ('t2pi_transformer', T2PITransformer()),
                ('classifier', MultinomialNB())])
```

[149]:
```
y_pred = t2pi_model.predict(X_test) #predict testing data

print(classification_report(y_test, y_pred))
```

```
transforming …
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=93.0),␣
 ↪HTML(value='')))
```

```
sparsity(X):
=> before 0.9958
=>  after 0.5103
```

	precision	recall	f1-score	support
0	0.74	0.67	0.70	30
1	0.77	0.91	0.83	22
2	0.72	0.68	0.70	19
3	0.77	0.77	0.77	22
accuracy			0.75	93
macro avg	0.75	0.76	0.75	93
weighted avg	0.75	0.75	0.75	93

2.0.3 Initialize models

[150]:
```
# normal model
count_model = get_model(stop_words=None, classifier=get_classifier())
count_sw_model = get_model(stop_words="english", classifier=get_classifier())
```

```
tfidf_model = get_model(tfidf=True, stop_words=None,␣
 ↪classifier=get_classifier())
tfidf_sw_model = get_model(tfidf=True, stop_words="english",␣
 ↪classifier=get_classifier())

# model
t2pi_count_model = get_model(use_t2pi=True, stop_words=None,␣
 ↪classifier=get_classifier())
t2pi_count_sw_model = get_model(use_t2pi=True, stop_words="english",␣
 ↪classifier=get_classifier())

t2pi_tfidf_model = get_model(tfidf=True, use_t2pi=True, stop_words=None,␣
 ↪classifier=get_classifier())
t2pi_tfidf_sw_model = get_model(tfidf=True, use_t2pi=True,␣
 ↪stop_words="english", classifier=get_classifier())

models = {
    "count_model": count_model,
    "count_sw_model": count_model,
    "tfidf_model": tfidf_model,
    "tfidf_sw_model": tfidf_sw_model,
    "t2pi_count_model": t2pi_count_model,
    "t2pi_count_sw_model": t2pi_count_sw_model,
    "t2pi_tfidf_model": t2pi_tfidf_model,
    "t2pi_tfidf_sw_model": t2pi_tfidf_sw_model
}
```

2.0.4 Running Cross validation on all Models

[151]:
```
split_size = 5
skf = StratifiedKFold(n_splits=split_size, shuffle=True, random_state=100)

index = 0
macro_f1_scores, weighted_f1_scores, accuracies = [], [], []

for train_index, test_index in skf.split(X, y):
    index += 1

    x_train_fold, x_test_fold = X.iloc[train_index], X.iloc[test_index]
    y_train_fold, y_test_fold = y.iloc[train_index], y.iloc[test_index]

    accuracies.append([])
    macro_f1_scores.append([])
    weighted_f1_scores.append([])

    for model_name, model in models.items():
```

```
        print(f'-> {index}. {model_name} \n{"="*100}\n')
        model.fit(x_train_fold, y_train_fold)
        y_pred = model.predict(x_test_fold)

        accuracy = accuracy_score(y_test_fold, y_pred)
        weighted_f1_score = calculate_f1_score(y_test_fold, y_pred,␣
↪average='weighted')
        macro_f1_score = calculate_f1_score(y_test_fold, y_pred,␣
↪average='macro')

        weighted_f1_scores[-1].append(weighted_f1_score)
        macro_f1_scores[-1].append(macro_f1_score)
        accuracies[-1].append(accuracy)
```

```
-> 1. count_model
====================================================================================
====================

-> 1. count_sw_model
====================================================================================
====================

-> 1. tfidf_model
====================================================================================
====================

-> 1. tfidf_sw_model
====================================================================================
====================

-> 1. t2pi_count_model
====================================================================================
====================

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4689.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=297.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9889
=>  after 0.0018
```

```
transforming …
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=75.0),␣
 ↪HTML(value='')))
```

```
sparsity(X):
=> before 0.9910
=>  after 0.0011

-> 1. t2pi_count_sw_model
================================================================================
====================

creating term-term co-occurence pr matrix
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4425.0),␣
 ↪HTML(value='')))
```

```
transforming …
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=297.0),␣
 ↪HTML(value='')))
```

```
sparsity(X):
=> before 0.9935
=>  after 0.5317

transforming …
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=75.0),␣
 ↪HTML(value='')))
```

```
sparsity(X):
=> before 0.9957
=>  after 0.4964

-> 1. t2pi_tfidf_model
================================================================================
====================

creating term-term co-occurence pr matrix
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4689.0),␣
 ↪HTML(value='')))
```

```
transforming …
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=297.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9889
=>  after 0.0018

transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=75.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9910
=>  after 0.0011

-> 1. t2pi_tfidf_sw_model
================================================================================
====================

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4425.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=297.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9935
=>  after 0.5317

transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=75.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9957
=>  after 0.4964

-> 2. count_model
================================================================================
====================
```

```
-> 2. count_sw_model
================================================================================
====================

-> 2. tfidf_model
================================================================================
====================

-> 2. tfidf_sw_model
================================================================================
====================

-> 2. t2pi_count_model
================================================================================
====================

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4678.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=297.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9889
=>  after 0.0021

transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=75.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9911
=>  after 0.0011

-> 2. t2pi_count_sw_model
================================================================================
====================

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4421.0),␣
 ↪HTML(value='')))
```

```
transforming …
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=297.0),␣
 ↪HTML(value='')))
```

```
sparsity(X):
=> before 0.9935
=>  after 0.5205

transforming …
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=75.0),␣
 ↪HTML(value='')))
```

```
sparsity(X):
=> before 0.9957
=>  after 0.5133

-> 2. t2pi_tfidf_model
================================================================================
====================

creating term-term co-occurence pr matrix
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4678.0),␣
 ↪HTML(value='')))
```

```
transforming …
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=297.0),␣
 ↪HTML(value='')))
```

```
sparsity(X):
=> before 0.9889
=>  after 0.0021

transforming …
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=75.0),␣
 ↪HTML(value='')))
```

```
sparsity(X):
=> before 0.9911
=>  after 0.0011

-> 2. t2pi_tfidf_sw_model
```

```
================================================================================
====================

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4421.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=297.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9935
=>  after 0.5205

transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=75.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9957
=>  after 0.5133

-> 3. count_model
================================================================================
====================

-> 3. count_sw_model
================================================================================
====================

-> 3. tfidf_model
================================================================================
====================

-> 3. tfidf_sw_model
================================================================================
====================

-> 3. t2pi_count_model
================================================================================
====================

creating term-term co-occurence pr matrix
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4820.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=298.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9891
=>  after 0.0018

transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=74.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9916
=>  after 0.0011

-> 3. t2pi_count_sw_model
================================================================================
====================

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4560.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=298.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9935
=>  after 0.5181

transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=74.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9960
=>  after 0.5226
```

```
-> 3. t2pi_tfidf_model
================================================================================
====================

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4820.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=298.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9891
=>  after 0.0018

transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=74.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9916
=>  after 0.0011

-> 3. t2pi_tfidf_sw_model
================================================================================
====================

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4560.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=298.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9935
=>  after 0.5181

transforming …
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=74.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9960
=>  after 0.5226

-> 4. count_model
================================================================================
====================

-> 4. count_sw_model
================================================================================
====================

-> 4. tfidf_model
================================================================================
====================

-> 4. tfidf_sw_model
================================================================================
====================

-> 4. t2pi_count_model
================================================================================
====================

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4720.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=298.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9890
=>  after 0.0018

transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=74.0),␣
 ↪HTML(value='')))


sparsity(X):
```

```
=> before 0.9912
=>  after 0.0013

-> 4. t2pi_count_sw_model
================================================================================
====================

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4459.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=298.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9935
=>  after 0.5107

transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=74.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9959
=>  after 0.5348

-> 4. t2pi_tfidf_model
================================================================================
====================

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4720.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=298.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9890
=>  after 0.0018
```

```
transforming …
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=74.0),␣
 ↪HTML(value='')))
```

```
sparsity(X):
=> before 0.9912
=>  after 0.0013
```

```
-> 4. t2pi_tfidf_sw_model
================================================================================
====================
```

```
creating term-term co-occurence pr matrix
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4459.0),␣
 ↪HTML(value='')))
```

```
transforming …
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=298.0),␣
 ↪HTML(value='')))
```

```
sparsity(X):
=> before 0.9935
=>  after 0.5107
```

```
transforming …
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=74.0),␣
 ↪HTML(value='')))
```

```
sparsity(X):
=> before 0.9959
=>  after 0.5348
```

```
-> 5. count_model
================================================================================
====================
```

```
-> 5. count_sw_model
================================================================================
====================
```

```
-> 5. tfidf_model
================================================================================
```

```
====================

-> 5. tfidf_sw_model
================================================================================
====================

-> 5. t2pi_count_model
================================================================================
====================

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4681.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=298.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9889
=>  after 0.0000

transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=74.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9910
=>  after 0.0027

-> 5. t2pi_count_sw_model
================================================================================
====================

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4418.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=298.0),␣
 ↪HTML(value='')))
```

```
sparsity(X):
=> before 0.9935
=>  after 0.5145

transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=74.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9957
=>  after 0.5282

-> 5. t2pi_tfidf_model
================================================================================
====================

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4681.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=298.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9889
=>  after 0.0000

transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=74.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9910
=>  after 0.0027

-> 5. t2pi_tfidf_sw_model
================================================================================
====================

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4418.0),␣
 ↪HTML(value='')))
```

```
transforming …
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=298.0),␣
 ↪HTML(value='')))
```

```
sparsity(X):
=> before 0.9935
=>  after 0.5145

transforming …
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=74.0),␣
 ↪HTML(value='')))
```

```
sparsity(X):
=> before 0.9957
=>  after 0.5282
```

[152]:
```
model_names = list(models.keys())

accuracy = pd.DataFrame(data=np.array(accuracies), columns=model_names)
weighted_f1_score = pd.DataFrame(data=np.array(weighted_f1_scores),␣
 ↪columns=model_names)
macro_f1_score = pd.DataFrame(data=np.array(macro_f1_scores),␣
 ↪columns=model_names)

accuracy.loc["mean"] = accuracy.mean(0)
weighted_f1_score.loc["mean"] = weighted_f1_score.mean(0)
macro_f1_score.loc["mean"] = macro_f1_score.mean(0)
```

[153]:
```
accuracy.head()
```

[153]:

	count_model	count_sw_model	tfidf_model	tfidf_sw_model	t2pi_count_model	t2pi_count_sw_model	t2pi_tfidf_model	t2pi_tfidf_sw_model
0	0.760000	0.760000	0.733333	0.773333	0.720000	0.733333	0.680000	0.720000
1	0.773333	0.773333	0.786667	0.826667	0.746667	0.800000	0.773333	0.773333
2	0.851351	0.851351	0.824324	0.878378	0.851351	0.878378	0.864865	0.837838
3	0.716216	0.716216	0.756757	0.810811	0.770270	0.783784	0.770270	0.797297
4	0.689189	0.689189	0.702703	0.797297	0.729730	0.797297	0.743243	0.797297

[154]:
```
weighted_f1_score.head()
```

[154]:

	count_model	count_sw_model	tfidf_model	tfidf_sw_model	t2pi_count_model	t2pi_count_sw_model	t2pi_tfidf_model	t2pi_tfidf_sw_model
0	0.760608	0.760608	0.735532	0.771758	0.722789	0.732650	0.682007	0.719097
1	0.771255	0.771255	0.784462	0.826944	0.746129	0.799633	0.773883	0.773883
2	0.852350	0.852350	0.826179	0.878378	0.850965	0.878932	0.865346	0.838889
3	0.717914	0.717914	0.761986	0.810039	0.772021	0.784762	0.773810	0.797998
4	0.688146	0.688146	0.698454	0.797008	0.729310	0.797436	0.743455	0.798095

[155]:
```
macro_f1_score.head()
```

[155]:

	count_model	count_sw_model	tfidf_model	tfidf_sw_model	t2pi_count_model	t2pi_count_sw_model	t2pi_tfidf_model	t2pi_tfidf_sw_model
0	0.760786	0.760786	0.736041	0.771970	0.723340	0.732967	0.682536	0.719592
1	0.771494	0.771494	0.784936	0.827443	0.745599	0.798681	0.773270	0.773270
2	0.852398	0.852398	0.827252	0.878655	0.851123	0.878655	0.865323	0.839026
3	0.717976	0.717976	0.761614	0.810526	0.771271	0.784647	0.772619	0.797788
4	0.687580	0.687580	0.697889	0.796024	0.729778	0.797309	0.743252	0.797675

[ ]:

[ ]: