term_term_pr_distr_and_inference

December 1, 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 scipy.stats import entropy as calculate_entropy

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

from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import MultinomialNB, GaussianNB, ComplementNB
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
```

[3]:
```
df = pd.DataFrame([[1, 2, 6],[3, 4, 7], [9, 3, 2]])
df
```

[3]:

	0	1	2
0	1	2	6
1	3	4	7
2	9	3	2

[4]:
```
def sum_weight(x, word_word_pr_distr_prime):
    return word_word_pr_distr_prime.apply(lambda y: x*y, axis=0).sum(0)
```

[5]:
```
df.apply(sum_weight, axis=1, args=(df,))
```

[5]:

	0	1	2
0	61	28	32
1	78	43	60
2	36	36	79

Utility functions

[40]:
```
## 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')

def get_lemmatizer():
    return 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,␣
↪stop_words="english"):
    count_vectorizer = CountVectorizer(stop_words=stop_words, binary=True)
    tfidf_transformer = TfidfTransformer() if tfidf else None
    t2pi_transformer = T2PITransformer() if use_t2pi else None

    # 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 _max_weight(x, pbar, word_word_pr_distr_prime):
        pbar.update(1)
        return word_word_pr_distr_prime.apply(lambda y: x*y, axis=0).max(0)

    @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)

    @staticmethod
    def _weighted_mean_weight(x, pbar, word_word_pr_distr_prime):
        pbar.update(1)
        xt = word_word_pr_distr_prime.apply(lambda y: x*y, axis=0)
        return (xt * (xt / xt.sum(0))).sum(0)

    def fit(self, X, y=None, **fit_params):
        print("creating term-term co-occurence pr matrix")
        self.word_word_pr_distr = pd.DataFrame(data=0.0, columns=X.columns,␣
 ↪index=X.columns)

        for term in tqdm(X.columns):
#             self.word_word_pr_distr[term] = X[X[term] > 0].sum(0) / X[term].
 ↪sum()
            self.word_word_pr_distr[term] = X[X[term] > 0].sum(0) / X.sum(0)
```

```
        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

[15]:
```
# 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
```

[16]:
```
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

[17]:
```
data = pd.DataFrame(
    data={
        "text":all_docs.data,
        "label":all_docs.target
    }
)

data.head()
```

[17]:

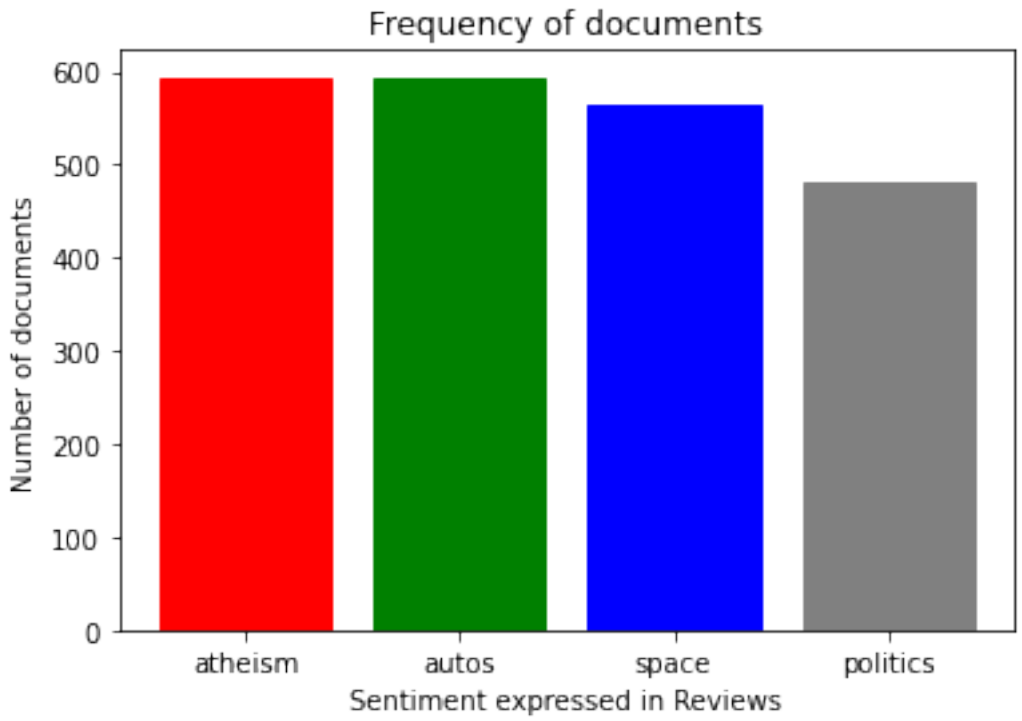
	text	label
0	\n\nI think that domestication will change beh…	0
1	\nI don't like this comment about "Typical" th…	3
2	\n<apparently you're not a woman - my husband …	1
3	While not exactly a service incident, I had a …	1
4	\n\nI think I can. Largely as a result of effo…	2

1.0.2 Label Frequency

[18]:
```
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

[19]:
```
max_size_per_class = 100

# remove long text
indices = data["text"].apply(data_isvalid, args=(lambda x: clean_text(x,␣
 ↪get_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()
```

[19]:

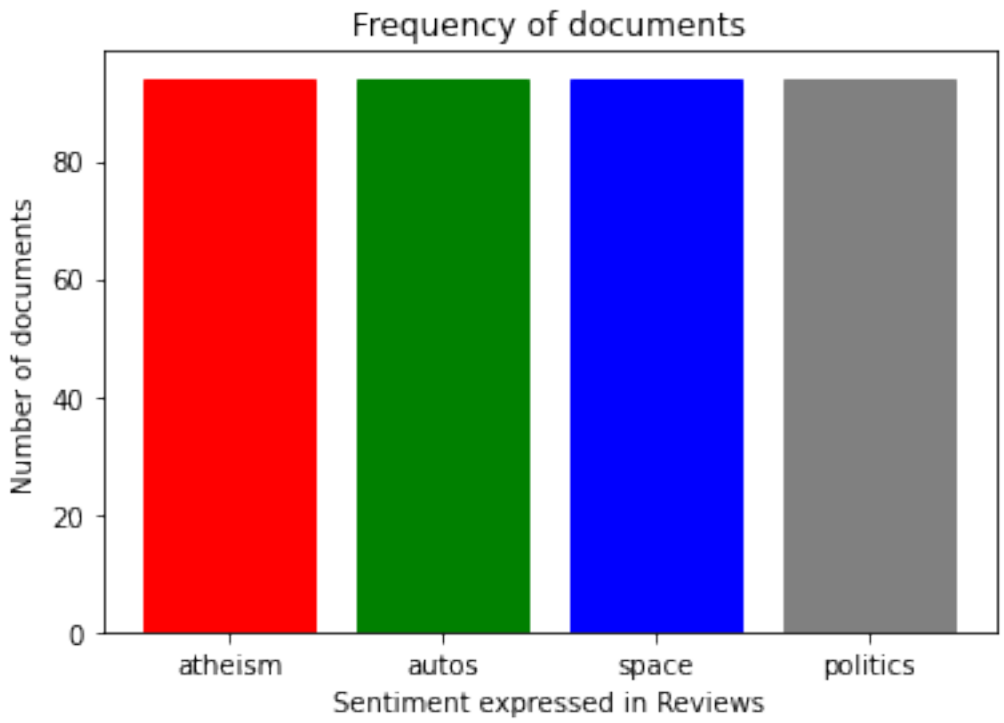
	text	label
0	\n\nI think that domestication will change beh…	0
30	\n[rest deleted…]\n\nYou were a liberal arts…	0
36	\nWorse? Maybe not, but it is definately a vi…	0
63	\nCould you expand on your definition of knowi…	0
65	\nLooking at historical evidence such 'perfect…	0

[20]:
```
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.
```

[21]:
```
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    94
2    94
1    94
0    94
Name: label, dtype: int64
```



2.0.1 initialize input and output

[22]:
```
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

[33]:
```
def get_classifier():
#     return MultinomialNB()
    return GaussianNB()
#     return ComplementNB()
```

[25]:
```
# initialize model
t2pi_model = get_model(use_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=4306.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=282.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9932
=>  after 0.5170
```

[25]:
```
Pipeline(steps=[('clean_text', CleanTextTransformer()),
                ('vectorizer',
                 CountVectorizer(binary=True, stop_words='english')),
                ('dense', DenseTransformer()),
                ('t2pi_transformer', T2PITransformer()),
                ('classifier', MultinomialNB())])
```

[26]:
```
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=94.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9954
=>  after 0.5086
```

	precision	recall	f1-score	support
0	0.75	0.80	0.77	30
1	0.62	0.83	0.71	18
2	0.94	0.76	0.84	21
3	0.81	0.68	0.74	25
accuracy			0.77	94
macro avg	0.78	0.77	0.77	94
weighted avg	0.78	0.77	0.77	94

2.0.3 Initialize models

[41]:
```
# 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

[42]:
```
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=4765.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=300.0),␣
 ↪HTML(value='')))
```

```
sparsity(X):
=> before 0.9890
=>  after 0.0015

transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=76.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9910
=>  after 0.0007

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

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4504.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=300.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9934
=>  after 0.5031

transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=76.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9957
=>  after 0.5114

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

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

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4765.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=300.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9890
=>  after 0.0015

transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=76.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9910
=>  after 0.0007

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

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4504.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=300.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9934
=>  after 0.5031

transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=76.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9957
=>  after 0.5114
```

```
-> 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=4644.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=301.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9887
=>  after 0.0015

transforming …

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


sparsity(X):
=> before 0.9911
=>  after 0.0009

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

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

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4386.0),␣
 ↪HTML(value='')))
```

```
transforming …
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=301.0),␣
 ↪HTML(value='')))
```

```
sparsity(X):
=> before 0.9933
=>  after 0.4986
```

```
transforming …
```

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

```
sparsity(X):
=> before 0.9958
=>  after 0.5145
```

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

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

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4644.0),␣
 ↪HTML(value='')))
```

```
transforming …
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=301.0),␣
 ↪HTML(value='')))
```

```
sparsity(X):
=> before 0.9887
=>  after 0.0015
```

```
transforming …
```

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

```
sparsity(X):
=> before 0.9911
=>  after 0.0009

-> 2. t2pi_tfidf_sw_model
================================================================================
====================

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4386.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=301.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9933
=>  after 0.4986

transforming …

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


sparsity(X):
=> before 0.9958
=>  after 0.5145

-> 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=4701.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=301.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9889
=>  after 0.0015

transforming …

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


sparsity(X):
=> before 0.9907
=>  after 0.0006

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

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4444.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=301.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9934
=>  after 0.5197

transforming …
```

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


sparsity(X):
=> before 0.9955
=>  after 0.4834

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

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4701.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=301.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9889
=>  after 0.0015

transforming …

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


sparsity(X):
=> before 0.9907
=>  after 0.0006

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

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4444.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=301.0),␣
 ↪HTML(value='')))
```

```
sparsity(X):
=> before 0.9934
=>  after 0.5197

transforming …
```

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

```
sparsity(X):
=> before 0.9955
=>  after 0.4834

-> 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=4732.0),␣
 ↪HTML(value='')))
```

```
transforming …
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=301.0),␣
 ↪HTML(value='')))
```

```
sparsity(X):
=> before 0.9889
=>  after 0.0014
```

```
transforming …
```

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

```
sparsity(X):
=> before 0.9911
=>  after 0.0015

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

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

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4472.0),␣
 ↪HTML(value='')))
```

```
transforming …
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=301.0),␣
 ↪HTML(value='')))
```

```
sparsity(X):
=> before 0.9934
=>  after 0.4966

transforming …
```

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

```
sparsity(X):
=> before 0.9957
=>  after 0.5169

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

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

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4732.0),␣
 ↪HTML(value='')))
```

```
transforming …
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=301.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9889
=>  after 0.0014

transforming …

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


sparsity(X):
=> before 0.9911
=>  after 0.0015

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

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4472.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=301.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9934
=>  after 0.4966

transforming …

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


sparsity(X):
=> before 0.9957
=>  after 0.5169

-> 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=4792.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=301.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9890
=>  after 0.0001

transforming …

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


sparsity(X):
=> before 0.9912
=>  after 0.0023

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

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4530.0),␣
 ↪HTML(value='')))
```

```
transforming …
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=301.0),␣
 ↪HTML(value='')))
```

```
sparsity(X):
=> before 0.9935
=>  after 0.5118

transforming …
```

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

```
sparsity(X):
=> before 0.9956
=>  after 0.5046

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

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

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4792.0),␣
 ↪HTML(value='')))
```

```
transforming …
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=301.0),␣
 ↪HTML(value='')))
```

```
sparsity(X):
=> before 0.9890
=>  after 0.0001

transforming …
```

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

```
sparsity(X):
=> before 0.9912
=>  after 0.0023

-> 5. t2pi_tfidf_sw_model
```

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

creating term-term co-occurence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=4530.0),␣
 ↪HTML(value='')))


transforming …

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=301.0),␣
 ↪HTML(value='')))


sparsity(X):
=> before 0.9935
=>  after 0.5118

transforming …

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


sparsity(X):
=> before 0.9956
=>  after 0.5046
```

[43]:
```
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)
```

[44]:
```
accuracy.head(split_size+1)
```

[44]:

	count_model	count_sw_model	tfidf_model	tfidf_sw_model	\
0	0.763158	0.763158	0.736842	0.750000	
1	0.693333	0.693333	0.613333	0.640000	
2	0.786667	0.786667	0.760000	0.746667	
3	0.733333	0.733333	0.720000	0.773333	
4	0.800000	0.800000	0.813333	0.786667	

```
mean     0.755298        0.755298     0.728702        0.739333

      t2pi_count_model  t2pi_count_sw_model  t2pi_tfidf_model  \
0             0.671053             0.631579          0.631579
1             0.626667             0.640000          0.600000
2             0.773333             0.693333          0.653333
3             0.653333             0.626667          0.613333
4             0.706667             0.680000          0.653333
mean          0.686211             0.654316          0.630316

      t2pi_tfidf_sw_model
0                0.578947
1                0.613333
2                0.600000
3                0.573333
4                0.680000
mean             0.609123
```

[45]:
```
weighted_f1_score.head(split_size+1)
```

[45]:
```
      count_model  count_sw_model  tfidf_model  tfidf_sw_model  \
0        0.764414        0.764414     0.737613        0.751326
1        0.694172        0.694172     0.612228        0.635940
2        0.787438        0.787438     0.763333        0.750339
3        0.732395        0.732395     0.720255        0.774131
4        0.801961        0.801961     0.814168        0.787935
mean     0.756076        0.756076     0.729519        0.739934

      t2pi_count_model  t2pi_count_sw_model  t2pi_tfidf_model  \
0             0.667628             0.625668          0.622745
1             0.622020             0.637925          0.593995
2             0.772054             0.690563          0.650671
3             0.656781             0.625715          0.615745
4             0.706890             0.676096          0.652571
mean          0.685075             0.651193          0.627146

      t2pi_tfidf_sw_model
0                0.568777
1                0.608514
2                0.599671
3                0.571451
4                0.675022
mean             0.604687
```

[46]:
```
macro_f1_score.head(split_size+1)
```

[46]:

	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	0.764414	0.764414	0.737613	0.751326	0.667628	0.625668	0.622745	0.568777
1	0.694284	0.694284	0.611996	0.635396	0.622607	0.637841	0.594715	0.608817
2	0.787199	0.787199	0.762500	0.749238	0.771398	0.690923	0.650645	0.599878
3	0.732261	0.732261	0.720066	0.774007	0.656628	0.625222	0.615383	0.571242
4	0.801277	0.801277	0.813982	0.787934	0.706360	0.675094	0.651959	0.673659
mean	0.755887	0.755887	0.729231	0.739580	0.684924	0.650950	0.627089	0.604475

[ ]:

[ ]: