

term\_term\_pr\_distr\_and\_inference-multinomial-min\_df-3-size-150-max

December 8, 2020

autoreload modules and utilities

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

import all necessary 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.pipeline import FeatureUnion
from sklearn.naive_bayes import MultinomialNB, GaussianNB
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

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

```
[3]: ## 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 clean_text(text: str, lemmatizer = lambda x: x) -> str:
    # removes upper cases
    text = text.lower().strip()

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

    #lemmatize the words and join back into string text
    text = " ".join([lemmatizer(word) for word in word_tokenize(text)])
    return text

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

def get_pipeline(vectorizer_type, classifier, use_t2pi, min_df=3,
    ↪stop_words=None, lemmatizer = lambda x: x):
    vectorizer = CountVectorizer if vectorizer_type == "count" else ↪
    ↪TfidfVectorizer
    models = [
        ('clean_text', CleanTextTransformer(lemmatizer)),

        ("vectorizers", FeatureUnion([
            ('count_binary', CountVectorizer(stop_words=stop_words, ↪
    ↪binary=True, min_df=min_df)),
            ("count", vectorizer(stop_words=stop_words, min_df=min_df))
        ])),
    ]

    if use_t2pi:
        models.append(('t2pi_transformer', T2PITransformer()))

    models.append(('classifier', classifier))
    return Pipeline(models)

```

```

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 T2PITransformer(TransformerMixin):
    @staticmethod
    def _max_weight(x, pbar, word_word_pr):
        pbar.update(1)
        return (word_word_pr.T * x).max(0)

    @staticmethod
    def _sum_weight(x, pbar, word_word_pr):
        pbar.update(1)
        return (word_word_pr.T * x).sum(0)

    def fit(self, X, y=None, **fit_params):
        X = X[:, :int(X.shape[1]/2)].toarray()

        print("creating term-term co-occurrence pr matrix")
        terms = np.arange(X.shape[1])

        X = pd.DataFrame(X, columns=terms)
        self.word_word_pr_distr = pd.DataFrame(data=0.0, columns=terms,
        ↪index=terms)

        for term in tqdm(terms):
            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):
        X = X[:, int(X.shape[1]/2):].toarray()
        X = pd.DataFrame(X, columns=self.word_word_pr_distr.columns)

```

```

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

        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

```

[4]: # 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

[5]: 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

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

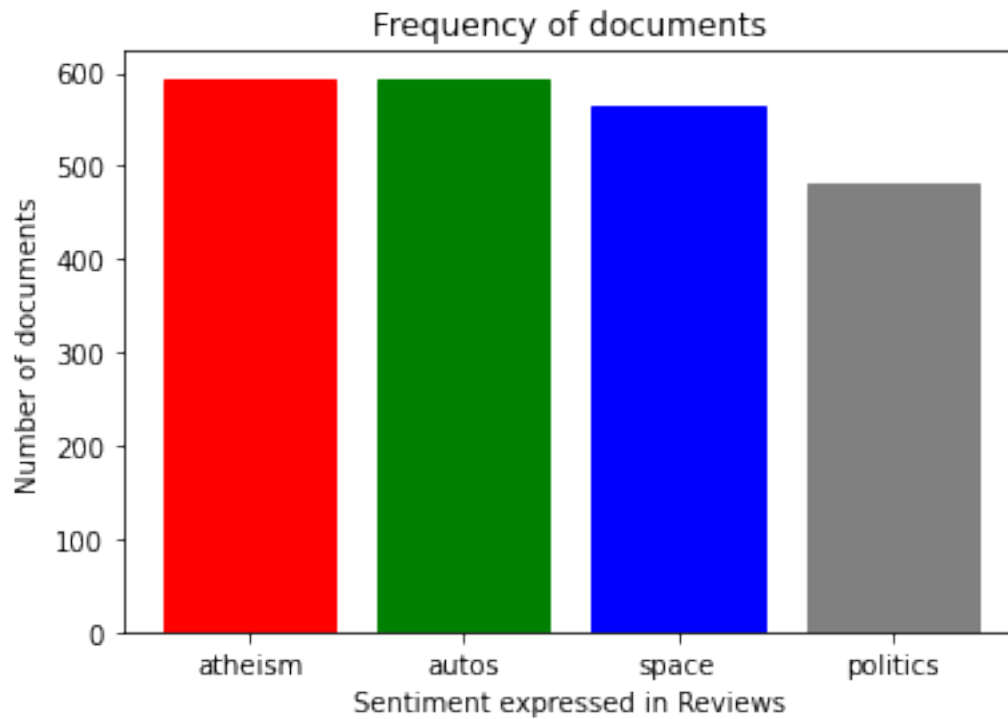
```
[6]:
```

	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

```
[7]: 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

### 1.0.3 Parameters

```
[8]: min_df = 3
stop_words = "english"

def get_classifier():
    # return GaussianNB()
    return MultinomialNB()

def get_lemmatizer():
    # return lambda x: x
    return WordNetLemmatizer().lemmatize
```

## 2 Select Valid Data

```
[9]: max_size_per_class = 150

# remove long text
indices = data["text"].apply(data_isvalid, args=(lambda x: clean_text(x,
    ↳get_lemmatizer()), 128, 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()

```

```

[9]:

```

	text	label
0	\n\nI think that domestication will change beh...	0
19	\n\n\tI agree, we spend too much energy on the...	0
30	\n[rest deleted...]\n\nYou were a liberal arts...	0
36	\nWorse? Maybe not, but it is definately a vi...	0
50	\n\n Could you explain what any of the above p...	0

```

[10]: 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.

```

[11]: print(data["label"].value_counts())
print()

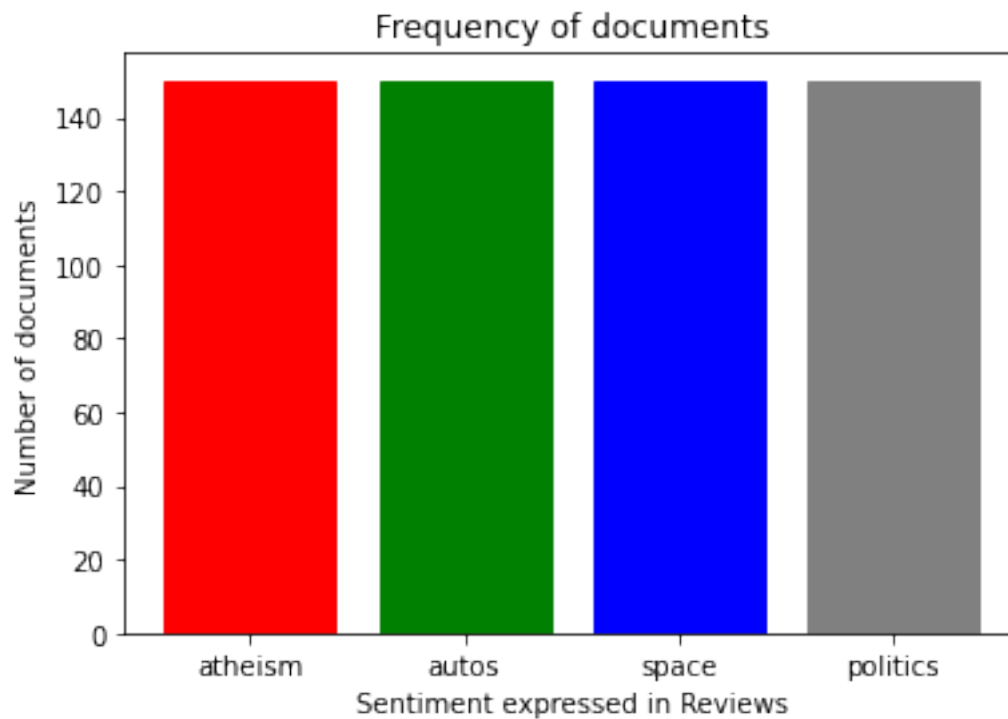
barlist = plt.bar(categories, data["label"].value_counts())

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    150
2    150
1    150
0    150
Name: label, dtype: int64
```



### 2.0.1 initialize input and output

```
[12]: 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

```
[13]: # initialize model
      t2pi_model = get_pipeline("count", get_classifier(), use_t2pi=True,
      ↪min_df=min_df, stop_words=None, lemmatizer = get_lemmatizer())
      t2pi_model
```



```
[13]: Pipeline(steps=[('clean_text', CleanTextTransformer()),
                        ('vectorizers',
                         FeatureUnion(transformer_list=[('count_binary',
                                                         CountVectorizer(binary=True,
                                                                       min_df=3)),
                                                         ('count',
                                                          CountVectorizer(min_df=3))])),
                        ('t2pi_transformer', T2PITransformer()),
                        ('classifier', MultinomialNB())])
```

```
[14]: # fit model
t2pi_model.fit(X_train, y_train)
```

creating term-term co-occurrence pr matrix

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1136.0),
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=450.0),
↳HTML(value='')))
```

```
[14]: Pipeline(steps=[('clean_text', CleanTextTransformer()),
                        ('vectorizers',
                         FeatureUnion(transformer_list=[('count_binary',
                                                         CountVectorizer(binary=True,
                                                                       min_df=3)),
                                                         ('count',
                                                          CountVectorizer(min_df=3))])),
                        ('t2pi_transformer', T2PITransformer()),
                        ('classifier', MultinomialNB())])
```

```
[15]: 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=150.0),
↳HTML(value='')))
```

	precision	recall	f1-score	support
0	0.71	0.67	0.69	45
1	0.76	0.78	0.77	37

2	0.75	0.56	0.64	32
3	0.63	0.81	0.71	36
accuracy			0.71	150
macro avg	0.71	0.70	0.70	150
weighted avg	0.71	0.71	0.70	150

### 2.0.3 Initialize models

```
[16]: # normal model
count_model = get_pipeline("count", get_classifier(), use_t2pi=False,
    ↳min_df=min_df, stop_words=None, lemmatizer = get_lemmatizer())
count_sw_model = get_pipeline("count", get_classifier(), use_t2pi=False,
    ↳min_df=min_df, stop_words=stop_words, lemmatizer = get_lemmatizer())

tfidf_model = get_pipeline("tfidf", get_classifier(), use_t2pi=False,
    ↳min_df=min_df, stop_words=None, lemmatizer = get_lemmatizer())
tfidf_sw_model = get_pipeline("tfidf", get_classifier(), use_t2pi=False,
    ↳min_df=min_df, stop_words=stop_words, lemmatizer = get_lemmatizer())

# model
t2pi_count_model = get_pipeline("count", get_classifier(), use_t2pi=True,
    ↳min_df=min_df, stop_words=None, lemmatizer = get_lemmatizer())
t2pi_count_sw_model = get_pipeline("count", get_classifier(), use_t2pi=True,
    ↳min_df=min_df, stop_words=stop_words, lemmatizer = get_lemmatizer())

t2pi_tfidf_model = get_pipeline("tfidf", get_classifier(), use_t2pi=True,
    ↳min_df=min_df, stop_words=None, lemmatizer = get_lemmatizer())
t2pi_tfidf_sw_model = get_pipeline("tfidf", get_classifier(), use_t2pi=True,
    ↳min_df=min_df, stop_words=stop_words, lemmatizer = get_lemmatizer())

models = {
    "count_model": count_model,
    "count_sw_model": count_sw_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

```
[17]: split_size = 7
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-occurrence pr matrix
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1273.0),
↳HTML(value='')))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=514.0),
↳HTML(value='')))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=86.0),
↳HTML(value='')))

-> 1. t2pi_count_sw_model
=====

creating term-term co-occurrence pr matrix
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1059.0),
↳HTML(value='')))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=514.0),
↳HTML(value='')))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=86.0),
↳HTML(value='')))

-> 1. t2pi_tfidf_model
=====

creating term-term co-occurrence pr matrix
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1273.0),
↳HTML(value='')))

```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=514.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=86.0),  
↳HTML(value='')))
```

-> 1. t2pi\_tfidf\_sw\_model

```
=====
```

creating term-term co-occurrence pr matrix

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1059.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=514.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=86.0),  
↳HTML(value='')))
```

-> 2. count\_model

```
=====
```

-> 2. count\_sw\_model

```
=====
```

-> 2. tfidf\_model

```
=====
```

-> 2. tfidf\_sw\_model

```
=====
```

-> 2. t2pi\_count\_model

```

=====
=====

creating term-term co-occurrence pr matrix
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1275.0),  

↳HTML(value=''))))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=514.0),  

↳HTML(value=''))))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=86.0),  

↳HTML(value=''))))

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

creating term-term co-occurrence pr matrix
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1056.0),  

↳HTML(value=''))))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=514.0),  

↳HTML(value=''))))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=86.0),  

↳HTML(value=''))))

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

creating term-term co-occurrence pr matrix
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1275.0),  

↳HTML(value=''))))

transforming ...

```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=514.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=86.0),  
↳HTML(value='')))
```

-> 2. t2pi\_tfidf\_sw\_model

```
=====
```

creating term-term co-occurrence pr matrix

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1056.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=514.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=86.0),  
↳HTML(value='')))
```

-> 3. count\_model

```
=====
```

-> 3. count\_sw\_model

```
=====
```

-> 3. tfidf\_model

```
=====
```

-> 3. tfidf\_sw\_model

```
=====
```

-> 3. t2pi\_count\_model

```
=====
```

creating term-term co-occurrence pr matrix

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1274.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=514.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=86.0),  
↳HTML(value='')))
```

-> 3. t2pi\_count\_sw\_model

```
=====
```

creating term-term co-occurrence pr matrix

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1057.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=514.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=86.0),  
↳HTML(value='')))
```

-> 3. t2pi\_tfidf\_model

```
=====
```

creating term-term co-occurrence pr matrix

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1274.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=514.0),  
↳HTML(value='')))
```



```

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=86.0),  

↳HTML(value=''))))

-> 3. t2pi_tfidf_sw_model
=====

creating term-term co-occurrence pr matrix
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1057.0),  

↳HTML(value=''))))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=514.0),  

↳HTML(value=''))))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=86.0),  

↳HTML(value=''))))

-> 4. count_model
=====

-> 4. count_sw_model
=====

-> 4. tfidf_model
=====

-> 4. tfidf_sw_model
=====

-> 4. t2pi_count_model
=====

creating term-term co-occurrence pr matrix

```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1274.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=514.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=86.0),  
↳HTML(value='')))
```

-> 4. t2pi\_count\_sw\_model

```
=====
```

creating term-term co-occurrence pr matrix

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1057.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=514.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=86.0),  
↳HTML(value='')))
```

-> 4. t2pi\_tfidf\_model

```
=====
```

creating term-term co-occurrence pr matrix

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1274.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=514.0),  
↳HTML(value='')))
```

```

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=86.0),  

↳HTML(value=''))))

-> 4. t2pi_tfidf_sw_model
=====

creating term-term co-occurrence pr matrix
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1057.0),  

↳HTML(value=''))))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=514.0),  

↳HTML(value=''))))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=86.0),  

↳HTML(value=''))))

-> 5. count_model
=====

-> 5. count_sw_model
=====

-> 5. tfidf_model
=====

-> 5. tfidf_sw_model
=====

-> 5. t2pi_count_model
=====

creating term-term co-occurrence pr matrix

```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1264.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=514.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=86.0),  
↳HTML(value='')))
```

-> 5. t2pi\_count\_sw\_model

```
=====
```

creating term-term co-occurrence pr matrix

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1044.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=514.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=86.0),  
↳HTML(value='')))
```

-> 5. t2pi\_tfidf\_model

```
=====
```

creating term-term co-occurrence pr matrix

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1264.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=514.0),  
↳HTML(value='')))
```

```

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=86.0),  

↳HTML(value=''))))

-> 5. t2pi_tfidf_sw_model
=====

creating term-term co-occurrence pr matrix
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1044.0),  

↳HTML(value=''))))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=514.0),  

↳HTML(value=''))))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=86.0),  

↳HTML(value=''))))

-> 6. count_model
=====

-> 6. count_sw_model
=====

-> 6. tfidf_model
=====

-> 6. tfidf_sw_model
=====

-> 6. t2pi_count_model
=====

creating term-term co-occurrence pr matrix

```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1292.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=515.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=85.0),  
↳HTML(value='')))
```

-> 6. t2pi\_count\_sw\_model

```
=====
```

creating term-term co-occurrence pr matrix

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1073.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=515.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=85.0),  
↳HTML(value='')))
```

-> 6. t2pi\_tfidf\_model

```
=====
```

creating term-term co-occurrence pr matrix

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1292.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=515.0),  
↳HTML(value='')))
```

```

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=85.0),  

↳HTML(value=''))))

-> 6. t2pi_tfidf_sw_model
=====

creating term-term co-occurrence pr matrix
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1073.0),  

↳HTML(value=''))))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=515.0),  

↳HTML(value=''))))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=85.0),  

↳HTML(value=''))))

-> 7. count_model
=====

-> 7. count_sw_model
=====

-> 7. tfidf_model
=====

-> 7. tfidf_sw_model
=====

-> 7. t2pi_count_model
=====

creating term-term co-occurrence pr matrix

```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1287.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=515.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=85.0),  
↳HTML(value='')))
```

-> 7. t2pi\_count\_sw\_model

```
=====
```

creating term-term co-occurrence pr matrix

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1069.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=515.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=85.0),  
↳HTML(value='')))
```

-> 7. t2pi\_tfidf\_model

```
=====
```

creating term-term co-occurrence pr matrix

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1287.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=515.0),  
↳HTML(value='')))
```



```
transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=85.0),
↳HTML(value=''))))

-> 7. t2pi_tfidf_sw_model
=====

creating term-term co-occurrence pr matrix
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1069.0),
↳HTML(value=''))))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=515.0),
↳HTML(value=''))))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=85.0),
↳HTML(value=''))))
```

```
[18]: 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)
```

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

```
[19]:
```

	count_model	count_sw_model	tfidf_model	tfidf_sw_model	\
0	0.790698	0.790698	0.790698	0.813953	
1	0.848837	0.848837	0.790698	0.813953	
2	0.767442	0.767442	0.720930	0.779070	
3	0.755814	0.755814	0.744186	0.744186	
4	0.639535	0.639535	0.651163	0.755814	
5	0.705882	0.705882	0.717647	0.729412	

6	0.694118	0.694118	0.705882	0.694118
mean	0.743189	0.743189	0.731601	0.761501

	t2pi_count_model	t2pi_count_sw_model	t2pi_tfidf_model	\
0	0.779070	0.837209	0.790698	
1	0.848837	0.860465	0.837209	
2	0.755814	0.767442	0.755814	
3	0.744186	0.732558	0.732558	
4	0.639535	0.732558	0.662791	
5	0.729412	0.729412	0.729412	
6	0.670588	0.694118	0.694118	
mean	0.738206	0.764823	0.743228	

	t2pi_tfidf_sw_model
0	0.825581
1	0.848837
2	0.790698
3	0.720930
4	0.744186
5	0.764706
6	0.705882
mean	0.771546

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

```
[20]:
```

	count_model	count_sw_model	tfidf_model	tfidf_sw_model	\
0	0.793588	0.793588	0.792005	0.813405	
1	0.848491	0.848491	0.791508	0.813069	
2	0.765940	0.765940	0.717751	0.775940	
3	0.755232	0.755232	0.742546	0.744384	
4	0.633116	0.633116	0.644383	0.751040	
5	0.703328	0.703328	0.715738	0.730247	
6	0.693308	0.693308	0.704881	0.693997	
mean	0.741857	0.741857	0.729830	0.760297	

	t2pi_count_model	t2pi_count_sw_model	t2pi_tfidf_model	\
0	0.780646	0.838026	0.790735	
1	0.848610	0.860938	0.837316	
2	0.752918	0.762584	0.752351	
3	0.744013	0.731442	0.733177	
4	0.633116	0.731213	0.660776	
5	0.727592	0.729392	0.727805	
6	0.669074	0.694041	0.694385	
mean	0.736567	0.763948	0.742363	

	t2pi_tfidf_sw_model
0	0.825307

1	0.848837
2	0.787803
3	0.717978
4	0.739650
5	0.765708
6	0.705617
mean	0.770129

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

```
[21]:
```

	count_model	count_sw_model	tfidf_model	tfidf_sw_model	\
0	0.793105	0.793105	0.791674	0.813710	
1	0.848370	0.848370	0.791588	0.812626	
2	0.766720	0.766720	0.718401	0.775478	
3	0.754968	0.754968	0.742405	0.743709	
4	0.632754	0.632754	0.643963	0.751143	
5	0.702384	0.702384	0.714735	0.730082	
6	0.693096	0.693096	0.704661	0.693601	
mean	0.741628	0.741628	0.729632	0.760050	

	t2pi_count_model	t2pi_count_sw_model	t2pi_tfidf_model	\
0	0.780392	0.838588	0.791070	
1	0.848722	0.860613	0.837459	
2	0.753387	0.762010	0.752624	
3	0.743818	0.730341	0.733305	
4	0.632754	0.731071	0.660436	
5	0.726937	0.729081	0.726503	
6	0.669103	0.693838	0.694186	
mean	0.736445	0.763649	0.742226	

	t2pi_tfidf_sw_model
0	0.825896
1	0.848787
2	0.787701
3	0.716959
4	0.739606
5	0.765829
6	0.705514
mean	0.770042