

# term\_term\_pr\_distr\_and\_inference-multinomial-100

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]: z = np.random.randint(20, size=(3, 5))
      z
```

```
[3]: array([[ 7, 12, 19,  4,  2],
            [15, 16, 13,  3, 10],
            [18,  8,  5,  1, 11]])
```

```
[4]: np.array(map(lambda x: x, [1,2,3]))
```

```
[4]: array(<map object at 0x0000016386551880>, dtype=object)
```

```
[5]: ## 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 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(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_distr_prime):
        pbar.update(1)
        return word_word_pr_distr_prime.apply(lambda y: x*y, axis=0).max(1)

    @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(1)

    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 ...")

    # new_sparsity after transform
    sparsity_before = calculate_sparsity(X)

    with tqdm(total=X.shape[0]) as pbar:
        X = X.apply(self._max_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

```
[24]: # 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
```

```
[25]: 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

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

data.head()
```

```
[26]:
```

	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

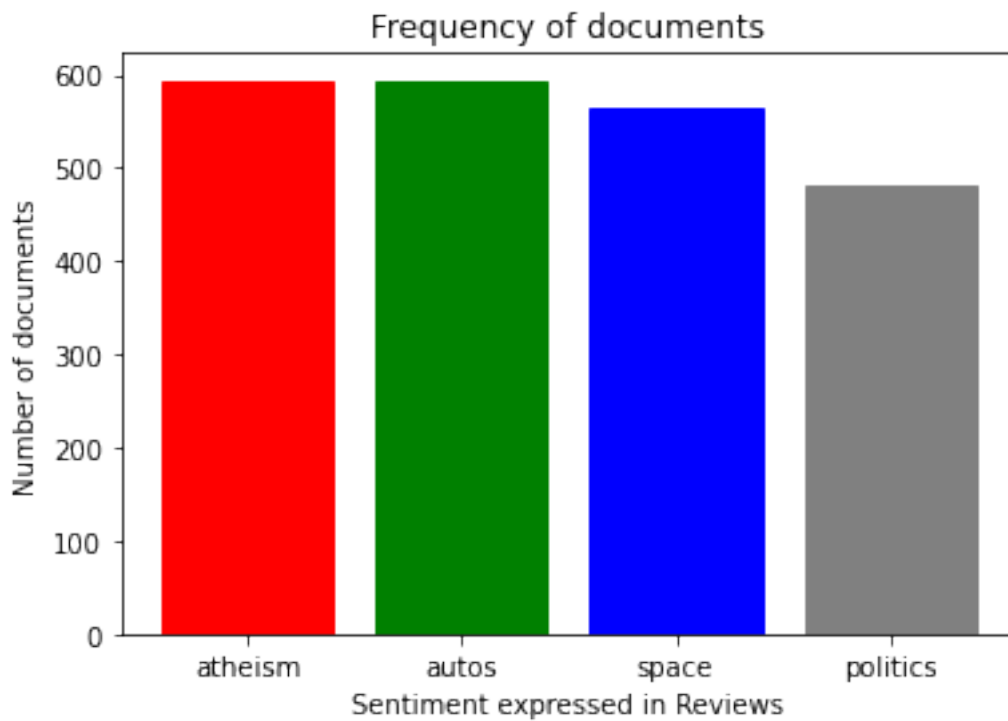
```
[27]: print(data["label"].value_counts())
print()

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

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

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

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

## 2 Select Valid Data

```
[29]: max_size_per_class = 100

# 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()
```

```
[29]:
```

	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

```
[30]: 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.

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

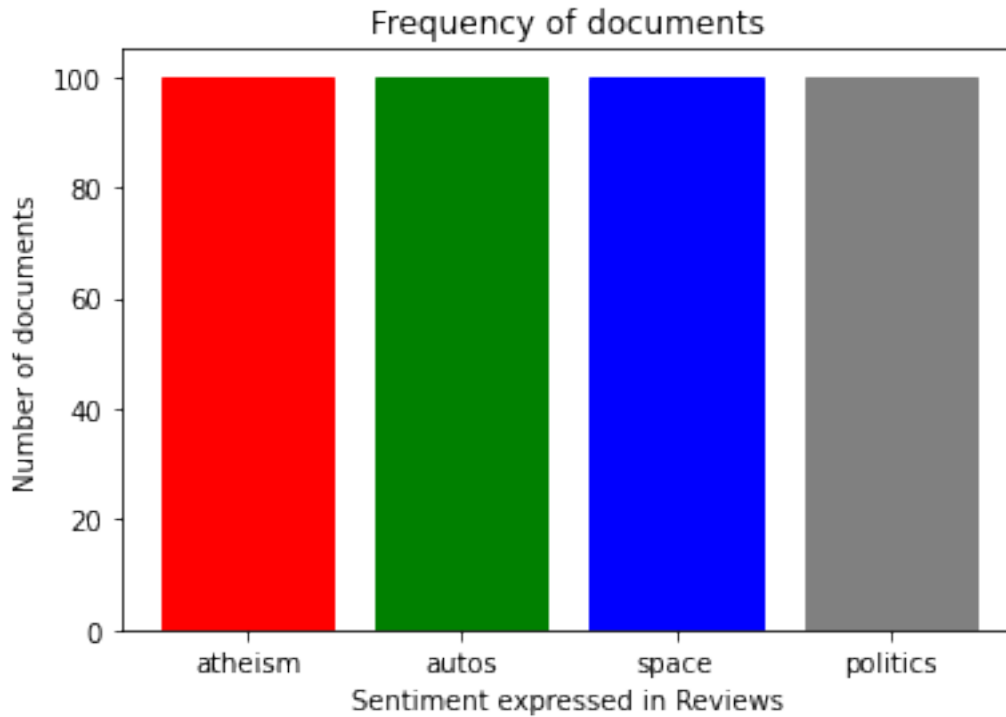
      barlist = plt.bar(categories, data["label"].value_counts())

      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      100
2      100
1      100
0      100
Name: label, dtype: int64
```





### 2.0.1 initialize input and output

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

```
[33]: 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())])
```

```
[34]: # 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=786.0),
↳HTML(value='')))
```

transforming ...

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

```
sparsity(X):
=> before 0.9620
=> after 0.9620
```

```
[34]: 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())])
```

```
[35]: 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), HTML(value='')))
```

```
sparsity(X):
=> before 0.9645
=> after 0.9645
```

	precision	recall	f1-score	support
0	0.64	0.58	0.61	31
1	0.54	0.67	0.60	21
2	0.62	0.60	0.61	25

	3	0.50	0.48	0.49	23
accuracy				0.58	100
macro avg		0.58	0.58	0.58	100
weighted avg		0.58	0.58	0.58	100

### 2.0.3 Initialize models

```
[36]: # 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

```
[37]: 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-occurrence pr matrix
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=802.0),
↳HTML(value='')))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=320.0),
↳HTML(value='')))

sparsity(X):
=> before 0.9629
=> after 0.9629

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=80.0),
↳HTML(value='')))

sparsity(X):
=> before 0.9642
=> after 0.9642

-> 1. t2pi_count_sw_model
=====

creating term-term co-occurrence pr matrix
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=611.0),
↳HTML(value='')))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=320.0),
↳HTML(value='')))

sparsity(X):
=> before 0.9820
=> after 0.9820

transforming ...

```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=80.0),  
↳HTML(value='')))
```

```
sparsity(X):  
=> before 0.9852  
=> after 0.9852
```

```
-> 1. t2pi_tfidf_model
```

```
=====
```

```
creating term-term co-occurrence pr matrix
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=802.0),  
↳HTML(value='')))
```

```
transforming ...
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=320.0),  
↳HTML(value='')))
```

```
sparsity(X):  
=> before 0.9629  
=> after 0.9629
```

```
transforming ...
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=80.0),  
↳HTML(value='')))
```

```
sparsity(X):  
=> before 0.9642  
=> after 0.9642
```

```
-> 1. t2pi_tfidf_sw_model
```

```
=====
```

```
creating term-term co-occurrence pr matrix
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=611.0),  
↳HTML(value='')))
```

```
transforming ...
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=320.0),  
↳HTML(value='')))
```

```
sparsity(X):
=> before 0.9820
=> after 0.9820
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=80.0),  
↳HTML(value='')))
```

```
sparsity(X):
=> before 0.9852
=> after 0.9852
```

```
-> 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=838.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=320.0),  
↳HTML(value='')))
```

```
sparsity(X):
=> before 0.9638
=> after 0.9638
```

```

transforming ...

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=80.0),  

↳HTML(value=''))))

sparsity(X):
=> before 0.9671
=> after 0.9671

-> 2. t2pi_count_sw_model
=====

creating term-term co-occurrence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=642.0),  

↳HTML(value=''))))

transforming ...

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=320.0),  

↳HTML(value=''))))

sparsity(X):
=> before 0.9826
=> after 0.9826

transforming ...

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=80.0),  

↳HTML(value=''))))

sparsity(X):
=> before 0.9853
=> after 0.9853

-> 2. t2pi_tfidf_model
=====

creating term-term co-occurrence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=838.0),  

↳HTML(value=''))))

transforming ...

```



```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=320.0),  
↳HTML(value='')))
```

```
sparsity(X):  
=> before 0.9638  
=> after 0.9638
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=80.0),  
↳HTML(value='')))
```

```
sparsity(X):  
=> before 0.9671  
=> after 0.9671
```

-> 2. t2pi\_tfidf\_sw\_model

```
=====
```

creating term-term co-occurrence pr matrix

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=642.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=320.0),  
↳HTML(value='')))
```

```
sparsity(X):  
=> before 0.9826  
=> after 0.9826
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=80.0),  
↳HTML(value='')))
```

```
sparsity(X):  
=> before 0.9853  
=> after 0.9853
```

-> 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=834.0),
↳HTML(value='')))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=320.0),
↳HTML(value='')))

sparsity(X):
=> before 0.9634
=> after 0.9634

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=80.0),
↳HTML(value='')))

sparsity(X):
=> before 0.9678
=> after 0.9678

-> 3. t2pi_count_sw_model
=====

creating term-term co-occurrence pr matrix
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=634.0),
↳HTML(value='')))

```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=320.0),  
↳HTML(value='')))
```

sparsity(X):

=> before 0.9819

=> after 0.9819

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=80.0),  
↳HTML(value='')))
```

sparsity(X):

=> before 0.9873

=> after 0.9873

-> 3. t2pi\_tfidf\_model

```
=====
```

creating term-term co-occurrence pr matrix

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=834.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=320.0),  
↳HTML(value='')))
```

sparsity(X):

=> before 0.9634

=> after 0.9634

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=80.0),  
↳HTML(value='')))
```

sparsity(X):

=> before 0.9678

=> after 0.9678

-> 3. t2pi\_tfidf\_sw\_model

```

=====
=====

creating term-term co-occurrence pr matrix
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=634.0),
↳HTML(value='')))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=320.0),
↳HTML(value='')))

sparsity(X):
=> before 0.9819
=> after 0.9819

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=80.0),
↳HTML(value='')))

sparsity(X):
=> before 0.9873
=> after 0.9873

-> 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=833.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=320.0),  
↳HTML(value='')))
```

```
sparsity(X):  
=> before 0.9641  
=> after 0.9641
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=80.0),  
↳HTML(value='')))
```

```
sparsity(X):  
=> before 0.9649  
=> after 0.9649
```

-> 4. t2pi\_count\_sw\_model

```
=====
```

creating term-term co-occurrence pr matrix

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=640.0),  
↳HTML(value='')))
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=320.0),  
↳HTML(value='')))
```

```
sparsity(X):  
=> before 0.9826  
=> after 0.9826
```

transforming ...

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=80.0),  
↳HTML(value='')))
```

```
sparsity(X):  
=> before 0.9852  
=> after 0.9852
```

```

-> 4. t2pi_tfidf_model
=====

creating term-term co-occurrence pr matrix
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=833.0),
↳HTML(value='')))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=320.0),
↳HTML(value='')))

sparsity(X):
=> before 0.9641
=> after 0.9641

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=80.0),
↳HTML(value='')))

sparsity(X):
=> before 0.9649
=> after 0.9649

-> 4. t2pi_tfidf_sw_model
=====

creating term-term co-occurrence pr matrix
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=640.0),
↳HTML(value='')))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=320.0),
↳HTML(value='')))

sparsity(X):
=> before 0.9826
=> after 0.9826

transforming ...

```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=80.0),  
↳HTML(value='')))
```

```
sparsity(X):  
=> before 0.9852  
=> after 0.9852
```

```
-> 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=846.0),  
↳HTML(value='')))
```

```
transforming ...
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=320.0),  
↳HTML(value='')))
```

```
sparsity(X):  
=> before 0.9637  
=> after 0.9637
```

```
transforming ...
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=80.0),  
↳HTML(value='')))
```

```
sparsity(X):
```

```

=> before 0.9686
=> after 0.9686

-> 5. t2pi_count_sw_model
=====

creating term-term co-occurrence pr matrix
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=647.0),
↳HTML(value='')))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=320.0),
↳HTML(value='')))

sparsity(X):
=> before 0.9823
=> after 0.9823

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=80.0),
↳HTML(value='')))

sparsity(X):
=> before 0.9867
=> after 0.9867

-> 5. t2pi_tfidf_model
=====

creating term-term co-occurrence pr matrix
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=846.0),
↳HTML(value='')))

transforming ...
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=320.0),
↳HTML(value='')))

sparsity(X):
=> before 0.9637
=> after 0.9637

```



```

transforming ...

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=80.0),
↳HTML(value='')))

sparsity(X):
=> before 0.9686
=> after 0.9686

-> 5. t2pi_tfidf_sw_model
=====

creating term-term co-occurrence pr matrix

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=647.0),
↳HTML(value='')))

transforming ...

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=320.0),
↳HTML(value='')))

sparsity(X):
=> before 0.9823
=> after 0.9823

transforming ...

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=80.0),
↳HTML(value='')))

sparsity(X):
=> before 0.9867
=> after 0.9867

```

```

[38]: 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)
```

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

```
[39]:
```

	count_model	count_sw_model	tfidf_model	tfidf_sw_model	\
0	0.6375	0.6375	0.5875	0.7375	
1	0.6375	0.6375	0.6250	0.6375	
2	0.6625	0.6625	0.6625	0.6875	
3	0.6875	0.6875	0.7000	0.7375	
4	0.5375	0.5375	0.5125	0.5750	
mean	0.6325	0.6325	0.6175	0.6750	

	t2pi_count_model	t2pi_count_sw_model	t2pi_tfidf_model	\
0	0.6500	0.7250	0.6875	
1	0.6625	0.6250	0.6500	
2	0.6500	0.6750	0.6500	
3	0.7000	0.7875	0.6875	
4	0.5375	0.5875	0.5500	
mean	0.6400	0.6800	0.6450	

	t2pi_tfidf_sw_model
0	0.7375
1	0.6250
2	0.7125
3	0.7750
4	0.6000
mean	0.6900

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

```
[40]:
```

	count_model	count_sw_model	tfidf_model	tfidf_sw_model	\
0	0.634565	0.634565	0.585685	0.733217	
1	0.639268	0.639268	0.624410	0.633373	
2	0.658217	0.658217	0.657095	0.685195	
3	0.691322	0.691322	0.706019	0.743601	
4	0.535197	0.535197	0.511022	0.572718	
mean	0.631714	0.631714	0.616846	0.673621	

	t2pi_count_model	t2pi_count_sw_model	t2pi_tfidf_model	\
0	0.642970	0.721312	0.685958	
1	0.660150	0.621789	0.648253	
2	0.644049	0.668560	0.646957	
3	0.706099	0.789868	0.694723	
4	0.533737	0.586582	0.552751	
mean	0.637401	0.677623	0.645728	

	t2pi_tfidf_sw_model
0	0.734565
1	0.618175
2	0.709398
3	0.778321
4	0.597289
mean	0.687550

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

	count_model	count_sw_model	tfidf_model	tfidf_sw_model	\
0	0.634565	0.634565	0.585685	0.733217	
1	0.639268	0.639268	0.624410	0.633373	
2	0.658217	0.658217	0.657095	0.685195	
3	0.691322	0.691322	0.706019	0.743601	
4	0.535197	0.535197	0.511022	0.572718	
mean	0.631714	0.631714	0.616846	0.673621	

	t2pi_count_model	t2pi_count_sw_model	t2pi_tfidf_model	\
0	0.642970	0.721312	0.685958	
1	0.660150	0.621789	0.648253	
2	0.644049	0.668560	0.646957	
3	0.706099	0.789868	0.694723	
4	0.533737	0.586582	0.552751	
mean	0.637401	0.677623	0.645728	

	t2pi_tfidf_sw_model
0	0.734565
1	0.618175
2	0.709398
3	0.778321
4	0.597289
mean	0.687550

```
[ ]:
```

```
[ ]:
```