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

```
[14]: ## 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
```

```python
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):
```

```python
        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()

        return self
```

```python
    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

```python
[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
```

```python
[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()
```
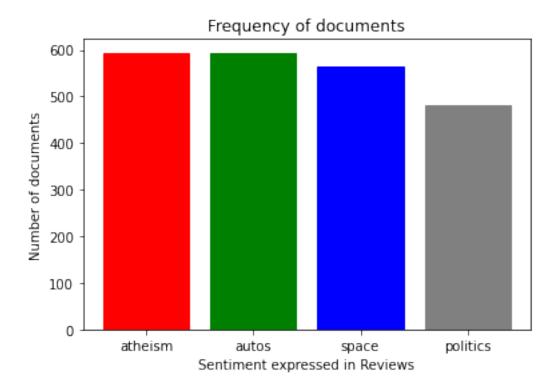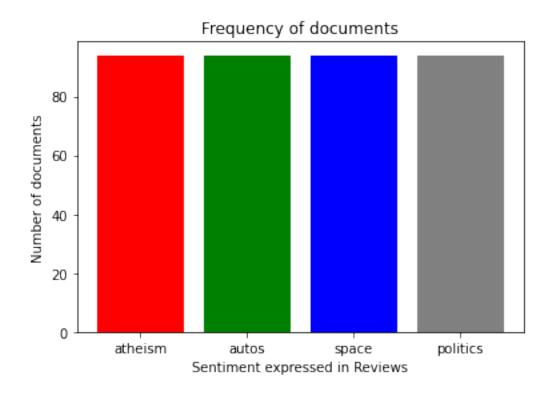
```
1    594
2    593
3    564
0    480
```

Name: label, dtype: int64

Frequency of documents

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]:
```python
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]:
```python
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
```

8

Frequency of documents

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

```
[24]: 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

```
[27]: # 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

```
[28]: 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
```

```python
[29]: 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)
```

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

```
[30]:      count_model  count_sw_model  tfidf_model  tfidf_sw_model  \
      0       0.763158        0.763158     0.763158        0.750000
      1       0.746667        0.746667     0.773333        0.813333
      2       0.786667        0.786667     0.760000        0.760000
      3       0.760000        0.760000     0.760000        0.720000
      4       0.840000        0.840000     0.840000        0.853333
```

```
mean        0.779298        0.779298        0.779298        0.779333


        t2pi_count_model  t2pi_count_sw_model  t2pi_tfidf_model  \
0            0.539474             0.605263          0.697368
1            0.586667             0.666667          0.693333
2            0.533333             0.706667          0.773333
3            0.506667             0.653333          0.706667
4            0.640000             0.813333          0.826667
mean         0.561228             0.689053          0.739474


        t2pi_tfidf_sw_model
0               0.710526
1               0.733333
2               0.746667
3               0.720000
4               0.853333
mean            0.752772
```

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

[31]:
```
        count_model  count_sw_model  tfidf_model  tfidf_sw_model  \
0          0.765498        0.765498     0.761620        0.743232
1          0.749175        0.749175     0.770699        0.811960
2          0.789240        0.789240     0.764865        0.760163
3          0.761350        0.761350     0.761350        0.719357
4          0.841083        0.841083     0.843347        0.853648
mean       0.781269        0.781269     0.780376        0.777672


        t2pi_count_model  t2pi_count_sw_model  t2pi_tfidf_model  \
0            0.538029             0.605065          0.697937
1            0.584931             0.670768          0.691792
2            0.532613             0.710761          0.773122
3            0.505332             0.650734          0.705675
4            0.638459             0.812522          0.827203
mean         0.559873             0.689970          0.739146


        t2pi_tfidf_sw_model
0               0.707419
1               0.731922
2               0.747867
3               0.719522
4               0.853220
mean            0.751990
```

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

```
[32]:         count_model   count_sw_model   tfidf_model   tfidf_sw_model  \
        0      0.765498         0.765498        0.761620         0.743232
        1      0.748764         0.748764        0.770946         0.812073
        2      0.789243         0.789243        0.765327         0.760119
        3      0.761859         0.761859        0.761859         0.719849
        4      0.840812         0.840812        0.842655         0.853212
        mean   0.781235         0.781235        0.780481         0.777697

               t2pi_count_model   t2pi_count_sw_model   t2pi_tfidf_model  \
        0            0.538029             0.605065            0.697937
        1            0.586681             0.670964            0.692384
        2            0.533428             0.710432            0.773618
        3            0.506202             0.651194            0.707058
        4            0.640180             0.812596            0.827084
        mean         0.560904             0.690050            0.739616

               t2pi_tfidf_sw_model
        0              0.707419
        1              0.731314
        2              0.747722
        3              0.719750
        4              0.853465
        mean           0.751934
```

[ ]:

[ ]: