TaskA_SVM

November 27, 2024

[1]:

```python
import numpy as np
import pandas as pd
# from datasets import load_dataset, Dataset, DatasetDict
from nltk.tokenize import sent_tokenize, word_tokenize
from gensim.models import Word2Vec
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.model_selection import KFold
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import confusion_matrix
from sklearn import svm
from tqdm import tqdm
from keras.preprocessing import sequence
from sklearn.decomposition import PCA
from nltk.probability import FreqDist
```

[2]:

```python
"""
Utility functions.
"""
def f1_score(tp, fp, fn):
    return (2 * tp) / (2 * tp + fp + fn)

def precision_score(tp, fp):
    return tp / (tp + fp)

def accuracy_score(tp, fp, tn, fn):
    return (tp + tn) / (tp + fp + tn + fn)

def recall_score(tp, fn):
    return tp / (tp + fn)

def flatten(matrix):
    flat_list = []
    for row in matrix:
        flat_list += row
    return flat_list
```

[3]:

```
"""
Download dataset SubtaskA.jsonl from
https://github.com/mbzuai-nlp/M4GT-Bench.
"""
DATA_PATH = "C:/Users/Admin/Desktop/cse847_proj/SubtaskA.jsonl"

# initialize dataset
df = pd.read_json(DATA_PATH, lines=True)
df = df[['text', 'label', 'model']]
print(df)
```

```
                                                    text  label   model
0       We consider a system of many polymers in solut…      1  cohere
1       We present a catalog of 66 YSOs in the Serpens…      1  cohere
2       Spectroscopic Observations of the Intermediate…      1  cohere
3       We present a new class of stochastic Lie group…      1  cohere
4       ALMA as the ideal probe of the solar chromosph…      1  cohere
…                                                    …    …     …
152804   The main results presented in this dissertati…      0   human
152805   Fine-grained sketch-based image retrieval (FG…      0   human
152806   We present the derivation of the NNLO two-par…      0   human
152807   The principle of optimism in the face of unce…      0   human
152808   We consider the setting of prediction with ex…      0   human

[152809 rows x 3 columns]
```

[14]:

```
"""
Evaluate model using count/TFIDF vectorization.
"""
def run_cv(model, X, y, count_vectorizer, tfidf_transformer=None):
    results = []
    k_fold = KFold(n_splits=K_FOLDS, shuffle=True, random_state=777)
    for train, test in tqdm(k_fold.split(X, y)):
        # split fold into training & testing sets
        X_train, y_train, X_test, y_test = X[train], y[train], X[test], y[test]

        # fit & transform data sets
        print("Count vectorizing...")
        X_train = count_vectorizer.fit_transform(X_train)
        X_test = count_vectorizer.transform(X_test)

        if tfidf_transformer:
            print("TFIDF transforming...")
            X_train = tfidf_transformer.fit_transform(X_train)
            X_test = tfidf_transformer.transform(X_test)

        # train the model
        print("Fitting the model...")
```

```python
        model.fit(X_train, y_train)

        # test the model
        print("Predicting the model...")
        y_hat = model.predict(X_test)

        # evaluate the model
        tn, fp, fn, tp = confusion_matrix(y_test, y_hat).ravel()
        results.append({
            'accuracy': accuracy_score(tp=tp, fp=fp, tn=tn, fn=fn),
            'recall': recall_score(tp=tp, fn=fn),
            'precision': precision_score(tp=tp, fp=fp),
            'f1': f1_score(tp=tp, fp=fp, fn=fn),
        })

    # analyze the run results
    results_df = pd.DataFrame.from_records(results).mean()

    return results_df
```

[15]:

```python
"""
Train and evaluate SVM classifier model using count/TFIDF vectorization.
"""
# consts
MAX_FEATURES = 3000
K_FOLDS = 3
MIN_DF = 2
MAX_DF = 0.7
NGRAM_RANGE = (1, 1)
ANALYZER = 'word'

# init model
model = svm.SVC(
    verbose=True,
    max_iter=-1,
    kernel='linear',
)

# load the data set
X = np.array(df.text)
y = np.array(df.label)

# init vectorizer and transformer
count_vectorizer = CountVectorizer(
    min_df=MIN_DF,
    max_df=MAX_DF,
    max_features=MAX_FEATURES,
    tokenizer=word_tokenize,
    token_pattern=None,
    ngram_range=NGRAM_RANGE,
    # strip_accents=STRIP_ACCENTS,
    # stop_words=STOP_WORDS,
)
tfidf_transformer = TfidfTransformer()

# run cross validation
results = run_cv(model, X, y, count_vectorizer, tfidf_transformer)
print(f"# model={model}, k_folds={K_FOLDS}, max_features={MAX_FEATURES},␣
 ↪min_df={MIN_DF}, max_df={MAX_DF}, "
          f"ngram_range={NGRAM_RANGE}\n{results}")
```

```
0it [00:00, ?it/s]

Count vectorizing…
TFIDF transforming…
Fitting the model…
[LibSVM]…*…
…*…*.*
optimization finished, #iter = 116277
obj = -27890.634461, rho = 2.379204
nSV = 31663, nBSV = 29212
Total nSV = 31663
Predicting the model…

1it [1:22:04, 4924.60s/it]

Count vectorizing…
TFIDF transforming…
Fitting the model…
[LibSVM]…*…
…*…*.*
optimization finished, #iter = 122330
obj = -28099.448361, rho = 2.310642
nSV = 31863, nBSV = 29406
Total nSV = 31863
Predicting the model…

2it [2:44:41, 4943.49s/it]

Count vectorizing…
TFIDF transforming…
Fitting the model…
[LibSVM]…*…
…*…*.*
optimization finished, #iter = 115238
obj = -27973.743292, rho = 2.387817
nSV = 31719, nBSV = 29263
```

```
Total nSV = 31719
Predicting the model…

3it [4:06:20, 4926.93s/it]

# model=SVC(kernel='linear', verbose=True), k_folds=3, max_features=3000,
min_df=2, max_df=0.7, ngram_range=(1, 1)
accuracy     0.887768
recall       0.904635
precision    0.900156
f1           0.902390
dtype: float64
```

[ ]: