supervised_ML_identify_author

April 4, 2025

1 Supervised Machine Learning - Identify Author

[1]:
```
!pip install -q spacy
```

[2]:
```
import spacy
```

1.1 The Dataset

Our two datasets are constructed from two related works of 19th century American transcendentalism. These are both public domain.

1. Essays by Ralph Waldo Emerson by Ralph Waldo Emerson
2. Walden, and On The Duty Of Civil Disobedience by Henry David Thoreau

These two authors had different writing styles but shared more than their philosophical interests—they were neighbors in Concord, Massachusetts.

These two works are also similar in length when formatted as plain text.

We will use spaCy to segment each work into sections of roughly 3 to 5 sentences each, then build a datafrom of the text including a label of 'emerson' or 'thoreau', then shuffle and split that into train and test sets for training some machine learning models to classify them by predicting which author they are from and compare the results.

We will also preprocess text to remove stopwords,and perform lemmatization.

[3]:
```
emerson_txt_url = "https://www.gutenberg.org/ebooks/16643.txt.utf-8"
thoreau_txt_url = "https://www.gutenberg.org/ebooks/205.txt.utf-8"
```

[4]:
```
import requests
from pathlib import Path
```

[5]:
```
def download_file(url):
  local_filename = Path(url.split('/')[-1])
  result = requests.get(url)
  result.raise_for_status()
  with open(local_filename, "wb") as f:
      f.write(result.content)
  return local_filename
```

```
[6]: emerson_file = download_file(emerson_txt_url)
     thoreau_file = download_file(thoreau_txt_url)
```

```
[7]: !head -n 50 {emerson_file}
```

```
The Project Gutenberg eBook of Essays by Ralph Waldo Emerson

This ebook is for the use of anyone anywhere in the United States and
most other parts of the world at no cost and with almost no restrictions
whatsoever. You may copy it, give it away or re-use it under the terms
of the Project Gutenberg License included with this ebook or online
at www.gutenberg.org. If you are not located in the United States,
you will have to check the laws of the country where you are located
before using this eBook.

Title: Essays by Ralph Waldo Emerson

Author: Ralph Waldo Emerson

Editor: Edna Henry Lee Turpin

Release date: September 4, 2005 [eBook #16643]
                Most recently updated: April 29, 2022

Language: English

Credits: Curtis A. Weyant, Sankar Viswanathan and the Online Distributed
Proofreading Team


*** START OF THE PROJECT GUTENBERG EBOOK ESSAYS BY RALPH WALDO EMERSON ***




                                ESSAYS

                                  BY

                          RALPH WALDO EMERSON




                        Merrill's English Texts

                SELECTED AND EDITED, WITH INTRODUCTION
```

```
                  AND NOTES, BY EDNA H.L. TURPIN, AUTHOR
                  OF "STORIES FROM AMERICAN HISTORY,"
                 "CLASSIC FABLES," "FAMOUS PAINTERS," ETC.



                                 NEW YORK

                          CHARLES E. MERRILL CO.
```

[8]:
```
!head -n 50 {thoreau_file}
```

```
The Project Gutenberg eBook of Walden, and On The Duty Of Civil Disobedience

This ebook is for the use of anyone anywhere in the United States and
most other parts of the world at no cost and with almost no restrictions
whatsoever. You may copy it, give it away or re-use it under the terms
of the Project Gutenberg License included with this ebook or online
at www.gutenberg.org. If you are not located in the United States,
you will have to check the laws of the country where you are located
before using this eBook.

Title: Walden, and On The Duty Of Civil Disobedience

Author: Henry David Thoreau

Release date: January 1, 1995 [eBook #205]
                Most recently updated: September 19, 2024

Language: English

Credits: Judith Boss, and David Widger


*** START OF THE PROJECT GUTENBERG EBOOK WALDEN, AND ON THE DUTY OF CIVIL
DISOBEDIENCE ***
WALDEN




and




ON THE DUTY OF CIVIL DISOBEDIENCE
```

```
by Henry David Thoreau


cover


Contents


 WALDEN

 Economy
 Where I Lived, and What I Lived For
 Reading
```

[9]:
```
# Let's strip the frontmatter lines off the start of each file.
# remove each line preceding one that contains "START OF THE PROJECT GUTENBERG␣
↪EBOOK "
!grep -n "START OF THE PROJECT GUTENBERG EBOOK" {emerson_file}
!grep -n "START OF THE PROJECT GUTENBERG EBOOK" {thoreau_file}
```

```
25:*** START OF THE PROJECT GUTENBERG EBOOK ESSAYS BY RALPH WALDO EMERSON ***
23:*** START OF THE PROJECT GUTENBERG EBOOK WALDEN, AND ON THE DUTY OF CIVIL
DISOBEDIENCE ***
```

[10]:
```
def trim_frontmatter(filename):
  with open(filename) as f:
    lines = f.readlines()

  n_trim_lines = 0
  for i, line in enumerate(lines):
    if "START OF THE PROJECT GUTENBERG EBOOK" in line:
      n_trim_lines = i + 1
      break

  trimmed_lines = lines[n_trim_lines:]
  trimmed_content = '\n'.join(trimmed_lines)
  new_filename = f"trimmed_{filename}"
  with open(new_filename, "w") as f:
    f.write(trimmed_content)
  return new_filename
```

[11]:
```
trimmed_emerson_file = trim_frontmatter(emerson_file)
trimmed_thoreau_file = trim_frontmatter(thoreau_file)
```

[12]:
```
!head {trimmed_emerson_file}
!head {trimmed_thoreau_file}
```

```
                                    ESSAYS


WALDEN
```

[13]:
```
from collections import deque
from random import randint
import itertools
```

[14]:
```
nlp = spacy.load("en_core_web_sm")
```

[15]:
```
def segment_doc(filename):
  with open(filename) as f:
    text = f.read()
  doc = nlp(text)
  assert doc.has_annotation("SENT_START")

  sent_dq = deque()
  #it = doc.sents.__iter__()
  n = randint(3, 5)

  for sent in doc.sents:
    sent_dq.append(sent)
    if len(sent_dq) > n:
      sent_dq.popleft()
      snippet = " ".join(sent.text for sent in sent_dq)
      yield snippet
      n = randint(3, 5)
```

```
        sent_dq.clear()
```

[16]:
```
import pandas as pd
```

[17]:
```
def dataframe_from_file(file_path):
  segments = segment_doc(file_path)

  df = pd.DataFrame(segments, columns=["text"])
  return df
```

[18]:
```
emerson_df = dataframe_from_file(trimmed_emerson_file)
emerson_df.to_csv("emerson.csv")
emerson_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1064 entries, 0 to 1063
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   text    1064 non-null   object
dtypes: object(1)
memory usage: 8.4+ KB
```

[19]:
```
emerson_df.head()
```

[19]:
```
                                                text
0  The editors of the several volumes will be\n\n…
1  He was descended\n\nfrom a long line of New En…
2  After graduating from college he taught school…
3  Although his sermons were\n\nalways couched in…
4  "\n\n\n\nEmerson did not long remain a ministe…
```

[20]:
```
thoreau_df = dataframe_from_file(trimmed_thoreau_file)
thoreau_df.to_csv("thoreau.csv")
thoreau_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 847 entries, 0 to 846
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   text    847 non-null    object
dtypes: object(1)
memory usage: 6.7+ KB
```

[21]:
```
thoreau_df.head()
```

```
[21]:                                                 text
     0  I lived there two\n\nyears and two months. At …
     1  In most books, the _I_, or first person,\n\nis…
     2  As for the rest of my readers,\n\nthey will ac…
     3  The twelve labors of Hercules\n\nwere trifling…
     4  Why should they eat their sixty acres, when ma…
```

```
[22]: # combine and shuffle the datasets, using a consistent random seed.
      from sklearn.utils import shuffle

      d1 = emerson_df.copy()
      d1["label"] = "emerson"

      d2 = thoreau_df.copy()
      d2["label"] = "thoreau"

      combined_df = pd.concat([d1, d2])
      combined_df = shuffle(combined_df, random_state=7919)
      combined_df.to_csv("combined.csv")
      combined_df.info()
      combined_df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1911 entries, 153 to 77
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   text    1911 non-null   object
 1   label   1911 non-null   object
dtypes: object(2)
memory usage: 44.8+ KB
```

```
[22]:                                                     text    label
     153  During this fast they abstain from the\n\ngrat…  thoreau
     638  The life of man is a self-evolving circle,[696…  emerson
     66   But perhaps a man is not required to bury hims…  thoreau
     409  Fine manners[398] show\n\nthemselves formidabl…  emerson
     201  We must learn to reawaken and keep ourselves a…  thoreau
```

1.2 Now we have our dataset in combined.csv

```
[23]: import matplotlib.pyplot as plt
      import seaborn as sns
```

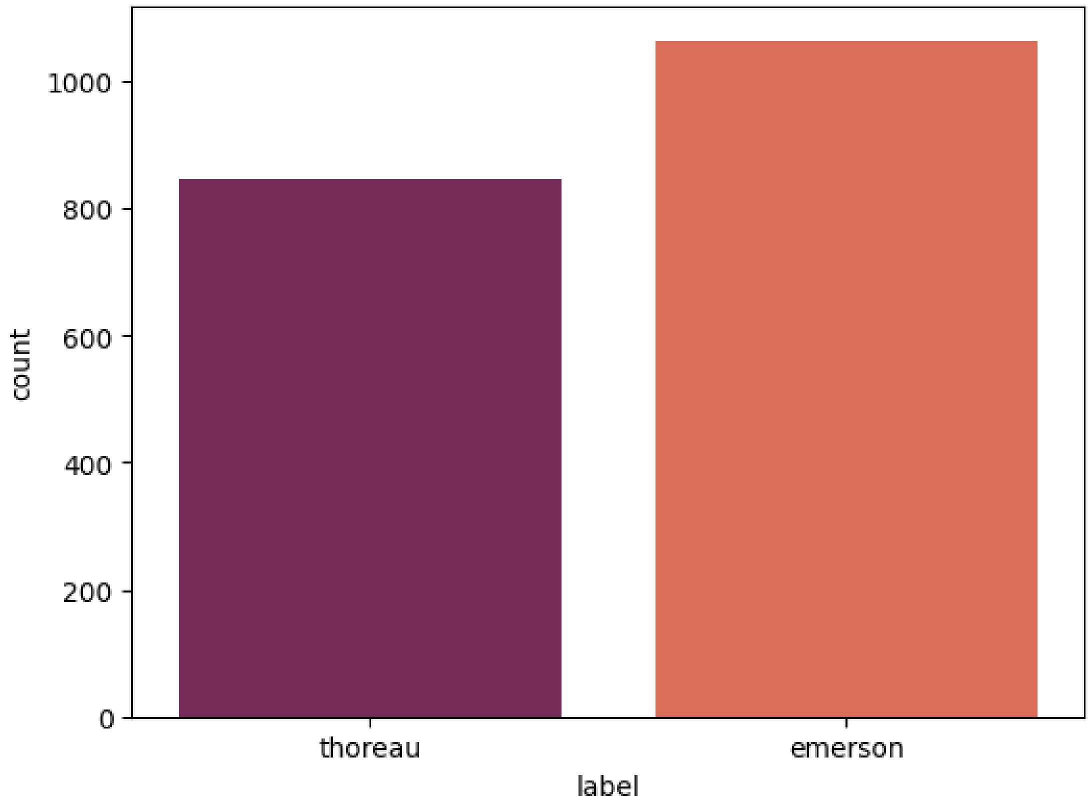
```
[24]: # you can start here if csv files were already created.
      from sklearn.feature_extraction.text import TfidfVectorizer
```

[25]:
```
sns.countplot(x=combined_df["label"], palette="rocket")
plt.show()
```

```
<ipython-input-25-33f8a76ce336>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(x=combined_df["label"], palette="rocket")
```



[26]:
```
!pip install -q wordcloud
```

[27]:
```
from spacy.lang.en import STOP_WORDS
my_stopwords = STOP_WORDS

', '.join(my_stopwords)
```

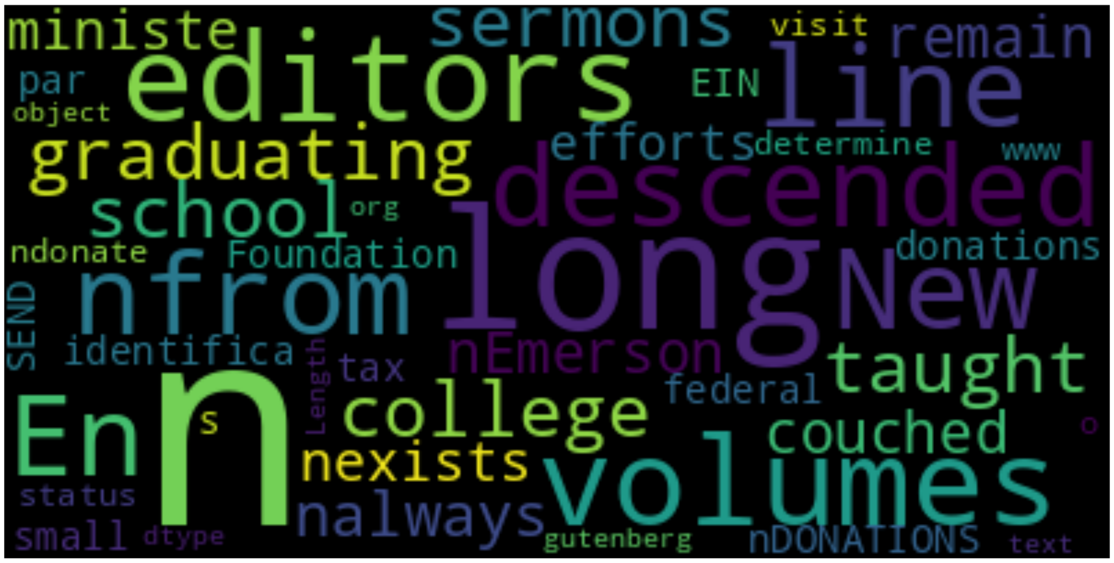
[27]:
```
"take, most, least, whereafter, both, they, might, same, will, below, say, his,
'm, 's, anyway, and, within, forty, before, eleven, due, it, there, seems, then,
call, go, n't, unless, 'll, nothing, fifty, whence, therefore, noone, rather,
amongst, either, serious, everywhere, on, enough, whereas, thence, somewhere,
front, whenever, whose, those, but, sometime, part, full, toward, thereafter,
have, we, became, becomes, together, although, seem, had, though, perhaps, five,
than, which, must, however, himself, ourselves, why, ‘m, our, did, own, its,
herein, them, everyone, without, somehow, into, such, side, ’d, the, give, at,
others, ’ve, put, only, just, where, several, my, whether, seemed, from, last,
've, well, behind, whole, you, ‘s, ‘ll, a, ‘re, third, each, seeming, too, ’s,
any, formerly, hereby, therein, itself, get, may, everything, thru, become, can,
once, out, up, by, name, 'd, many, beyond, all, even, afterwards, per, were,
alone, former, otherwise, around, always, nor, thereupon, doing, myself,
yourselves, thus, among, latter, becoming, yourself, sometimes, him, done, be,
elsewhere, very, is, whereupon, whither, to, ’m, upon, neither, across, their,
i, ’ll, themselves, re, does, indeed, often, more, hereafter, ’re, or, her,
someone, ca, being, so, anything, between, are, now, against, would, yours,
latterly, move, over, hence, this, make, none, hereupon, that, yet, was, some,
please, further, twenty, also, back, when, really, anyone, hundred, an, could,
here, if, since, me, as, these, via, whatever, two, herself, bottom, made,
never, beforehand, am, about, during, until, something, almost, what, above,
ten, still, us, four, ‘ve, because, besides, with, how, towards, your, 're, few,
sixty, six, down, along, keep, nowhere, not, amount, eight, n’t, already, ‘d,
throughout, other, been, much, do, fifteen, ever, no, first, every, see, nobody,
off, except, show, twelve, whoever, else, whom, cannot, one, anywhere, in, hers,
used, whereby, another, moreover, thereby, beside, anyhow, onto, should, next,
again, wherever, top, namely, empty, after, regarding, he, she, has, three, n't,
less, for, through, ours, nevertheless, various, of, under, quite, who, mostly,
using, wherein, while, nine, mine, meanwhile"
```

[28]:
```python
# Show wordcloud from each dataset.
from wordcloud import WordCloud


def plot_word_cloud(text_sections, title):
  cloud = WordCloud(background_color='black', stopwords=my_stopwords).
 ↪generate(str(text_sections))
  fig = plt.figure(figsize=(12,8), facecolor='white')
  plt.imshow(cloud, interpolation="bilinear")
  plt.axis('off')
  plt.title(title, fontsize=48)
  plt.tight_layout(pad=0)
  plt.show()
```

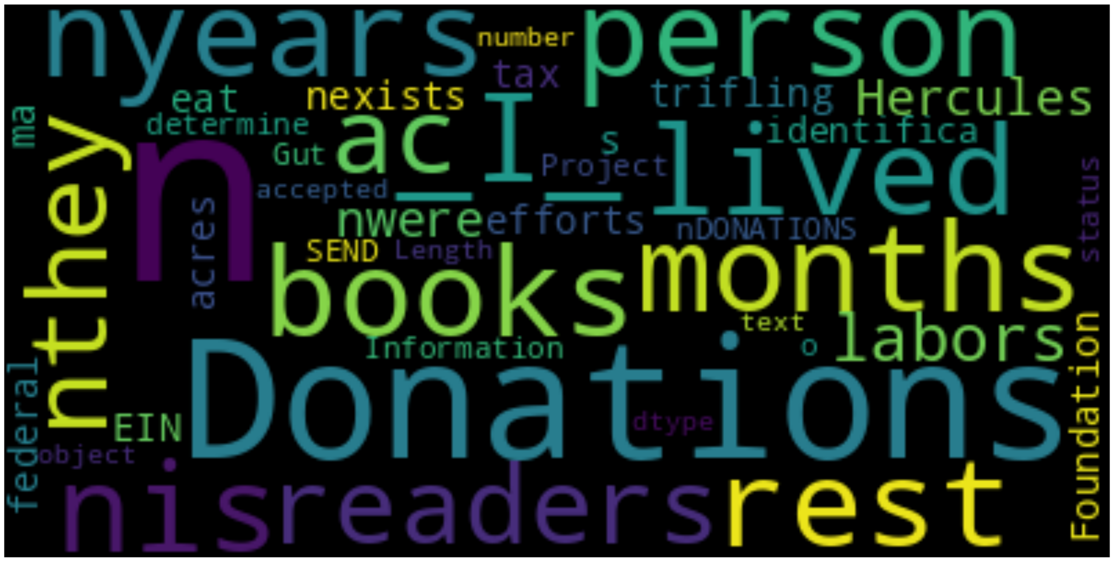
[29]:
```python
plot_word_cloud(emerson_df["text"], "Emerson")
```

Emerson



```
[30]: plot_word_cloud(thoreau_df["text"], "Thoreau")
```

Thoreau



```
[31]: # Preprocess text to remove stopwords, and perform lemmatization.

      final_text = []
      for index,entry in enumerate(combined_df['text']):
```

```
  doc = nlp(entry.lower())
  Final_words = []
  for word in doc:
    if not word.is_stop and not word.is_punct:
      Final_words.append(word.lemma_)
  final_text.append(' '.join(Final_words))
```

[32]:
```
combined_df['final_text'] = final_text
combined_df.head()
```

[32]:
```
                                                text    label  \
153  During this fast they abstain from the\n\ngrat…  thoreau
638  The life of man is a self-evolving circle,[696…  emerson
66   But perhaps a man is not required to bury hims…  thoreau
409  Fine manners[398] show\n\nthemselves formidabl…  emerson
201  We must learn to reawaken and keep ourselves a…  thoreau

                                          final_text
153  fast abstain \n\n gratification appetite passi…
638  life man self evolve circle,[696 \n\n ring imp…
66   man require bury \n\n  point important distinc…
409  fine manners[398 \n\n formidable uncultivated …
201  learn reawaken awake mechanical \n\n aid infin…
```

[33]:
```
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(combined_df["final_text"])
y = combined_df["label"]
```

[34]:
```
# split our data into train and test sets.
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=4909)
print(f"x_train: {x_train.shape}")
print(f"y_train: {y_train.shape}")
print(f"x_test: {x_test.shape}")
print(f"y_test: {y_test.shape}")
```

```
x_train: (1528, 11929)
y_train: (1528,)
x_test: (383, 11929)
y_test: (383,)
```

[35]:
```
from sklearn.linear_model import LogisticRegression
```

[36]:
```
# -2 for n_jobs is all but one CPU available.
lr_model = LogisticRegression(solver='saga', random_state=8102, n_jobs=-2)
```

```
lr_model.fit(x_train, y_train)
```

[36]:
```
LogisticRegression(n_jobs=-2, random_state=8102, solver='saga')
```

[37]:
```
from sklearn.metrics import f1_score

y_pred = lr_model.predict(x_test)
```

[38]:
```
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score
from IPython.display import Markdown, display

def show_metrics(y_test, y_pred, model_name):
  display(Markdown(f"# {model_name}"))

  print(classification_report(y_test,y_pred))
  print("Test accuracy:", accuracy_score(y_test,y_pred))
  cm = confusion_matrix(y_test, y_pred)

  labels = ["emerson", "thoreau"]
  sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels,␣
 ↪yticklabels=labels)
  plt.title('Confusion Matrix')
  plt.ylabel('Actual')
  plt.xlabel('Predicted')
  plt.show()
```

[39]:
```
show_metrics(y_test, y_pred, "Logistic Regression")
```

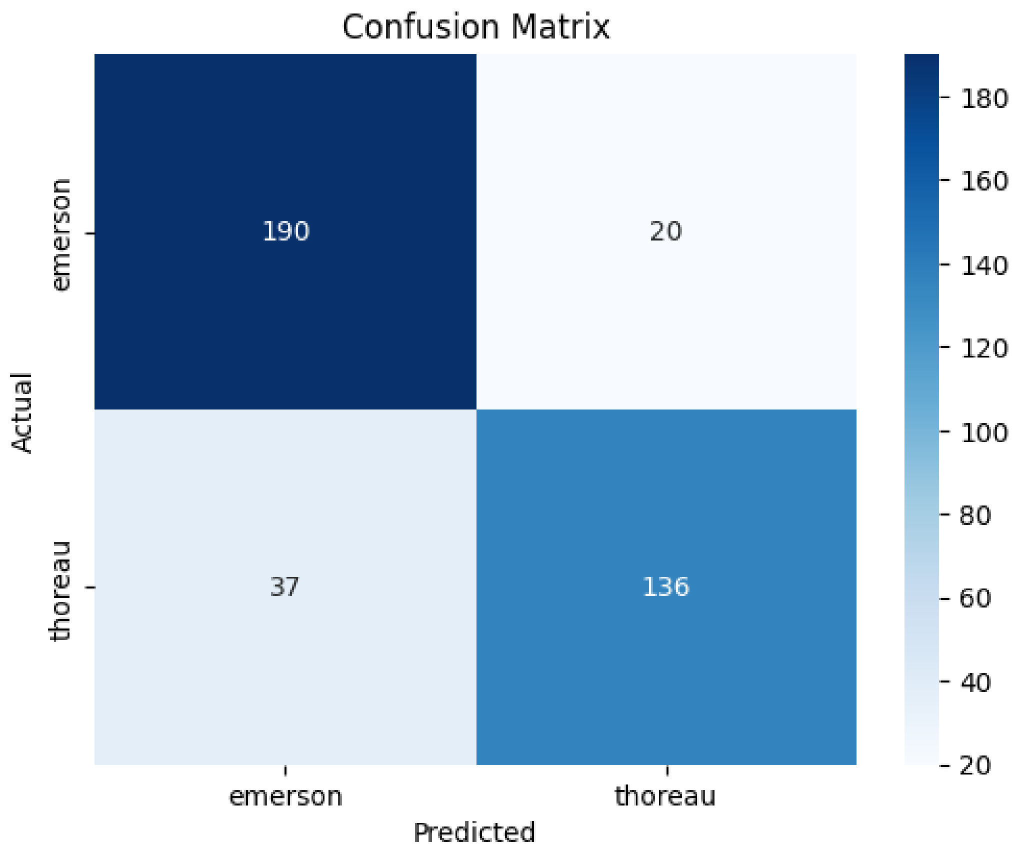
2 Logistic Regression

```
              precision    recall  f1-score   support

     emerson       0.84      0.90      0.87       210
     thoreau       0.87      0.79      0.83       173

    accuracy                           0.85       383
   macro avg       0.85      0.85      0.85       383
weighted avg       0.85      0.85      0.85       383

Test accuracy: 0.8511749347258486
```



[40]:
```
# Let's compare that to random forests.


from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()
rf.fit(x_train,y_train)
```

[40]:
```
RandomForestClassifier()
```

[41]:
```
y_pred_rf = rf.predict(x_test)
```

[42]:
```
show_metrics(y_test, y_pred_rf, "Random Forest")
```

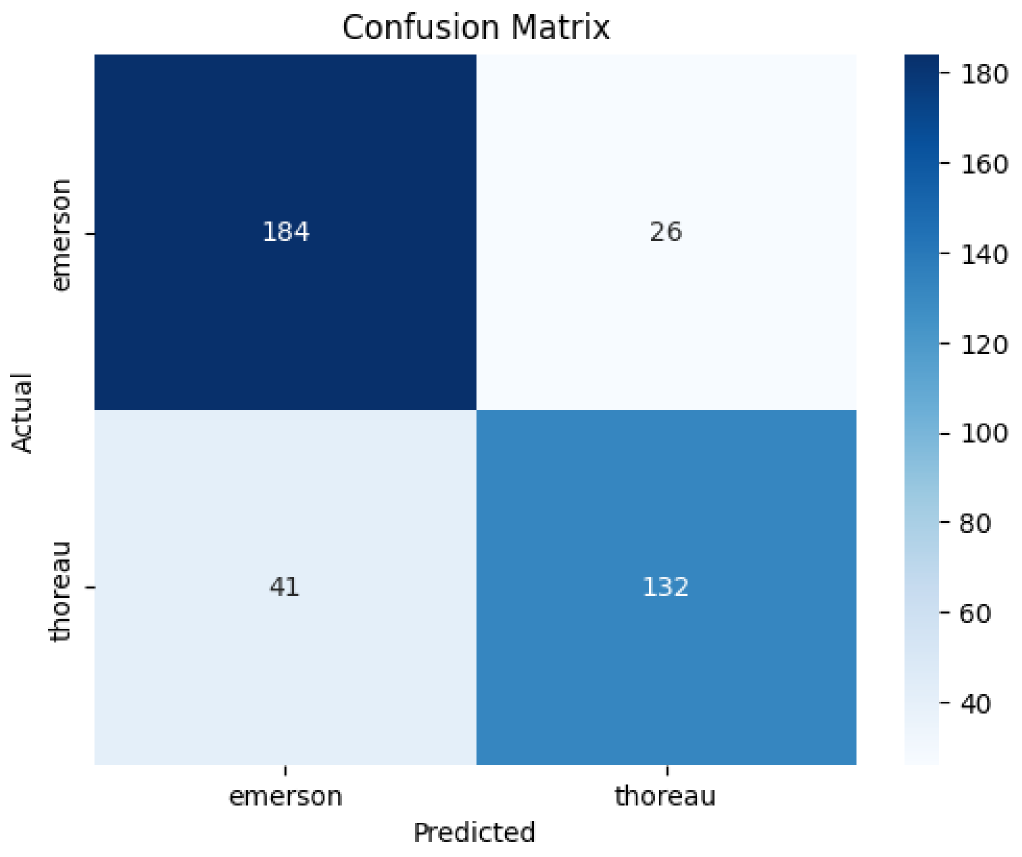
3 Random Forest

```
              precision    recall  f1-score   support

     emerson       0.82      0.88      0.85       210
     thoreau       0.84      0.76      0.80       173

    accuracy                           0.83       383
   macro avg       0.83      0.82      0.82       383
weighted avg       0.83      0.83      0.82       383

Test accuracy: 0.825065274151436
```



[43]:
```python
from sklearn import svm
```

[44]:
```python
# create the SVM classifier
clf = svm.SVC(kernel='rbf')

clf.fit(x_train,y_train)
clf
```

[44]:
```
SVC()
```

[45]:
```python
y_pred_svm = clf.predict(x_test)
```

```
[46]: show_metrics(y_test, y_pred_svm, "SVM")
```

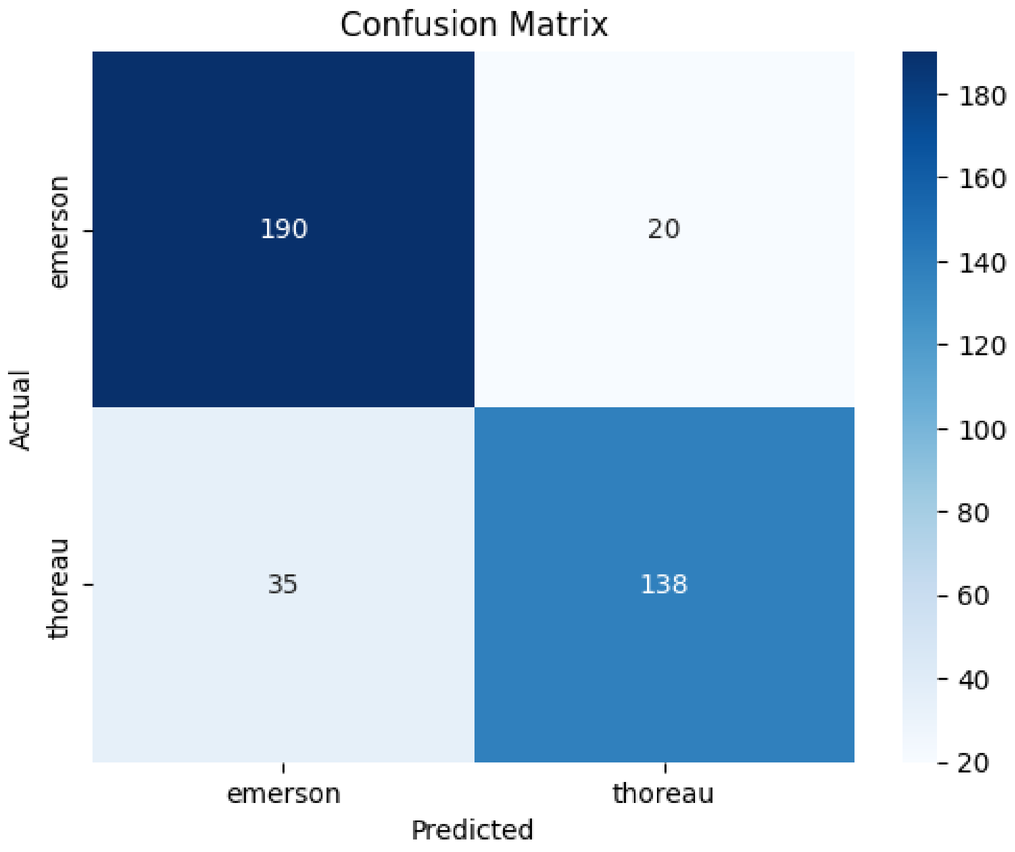
4 SVM

```
              precision    recall  f1-score   support

     emerson       0.84      0.90      0.87       210
     thoreau       0.87      0.80      0.83       173

    accuracy                           0.86       383
   macro avg       0.86      0.85      0.85       383
weighted avg       0.86      0.86      0.86       383

Test accuracy: 0.856396866840731
```



```
[47]: !pip install -q transformers
```

```
[48]: import torch
```

[49]:
```
from transformers import AutoTokenizer, AutoModel

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(device)
```

```
cuda
```

[50]:
```
tokenizer = AutoTokenizer.from_pretrained("distilbert-base-uncased",␣
  ↪return_token_type_ids = False, padding="max_length", truncation=True)

model = AutoModel.from_pretrained("distilbert-base-uncased").to(device)
```

```
tokenizer_config.json:   0%|          | 0.00/48.0 [00:00<?, ?B/s]

config.json:   0%|          | 0.00/483 [00:00<?, ?B/s]

vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/466k [00:00<?, ?B/s]

Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed.
Falling back to regular HTTP download. For better performance, install the
package with: `pip install huggingface_hub[hf_xet]` or `pip install hf_xet`
WARNING:huggingface_hub.file_download:Xet Storage is enabled for this repo, but
the 'hf_xet' package is not installed. Falling back to regular HTTP download.
For better performance, install the package with: `pip install
huggingface_hub[hf_xet]` or `pip install hf_xet`

model.safetensors:   0%|          | 0.00/268M [00:00<?, ?B/s]
```

[51]:
```
x_train_s, x_test_s, y_train_s, y_test_s =␣
  ↪train_test_split(combined_df["text"], combined_df["label"], test_size=0.2,␣
  ↪random_state=4909)
print(f"x_train_s: {x_train_s.shape}")
print(f"y_train_s: {y_train_s.shape}")
print(f"x_test_s: {x_test_s.shape}")
print(f"y_test_s: {y_test_s.shape}")
```

```
x_train_s: (1528,)
y_train_s: (1528,)
x_test_s: (383,)
y_test_s: (383,)
```

[52]:
```
x_train_tok = tokenizer(x_train_s.tolist(), padding=True, truncation=True,␣
  ↪return_tensors="pt")
y_train_tok = y_train_s.tolist()

x_test_tok = tokenizer(x_test_s.tolist(), padding=True, truncation=True,␣
  ↪return_tensors="pt")
y_test_tok = y_test_s.tolist()
```

[53]:
```
x_train_tok[0:2]
```

[53]:
```
[Encoding(num_tokens=512, attributes=[ids, type_ids, tokens, offsets,
attention_mask, special_tokens_mask, overflowing]),
 Encoding(num_tokens=512, attributes=[ids, type_ids, tokens, offsets,
attention_mask, special_tokens_mask, overflowing])]
```

[54]:
```
print(x_train_tok.keys())


#move onto device (GPU)
x_train_tok = {k:torch.tensor(v).to(device) for k,v in x_train_tok.items()}
x_test_tok = {k:torch.tensor(v).to(device) for k,v in x_test_tok.items()}
```

```
dict_keys(['input_ids', 'attention_mask'])
```

```
<ipython-input-54-cbedc1b8b59f>:5: UserWarning: To copy construct from a tensor,
it is recommended to use sourceTensor.clone().detach() or
sourceTensor.clone().detach().requires_grad_(True), rather than
torch.tensor(sourceTensor).
  x_train_tok = {k:torch.tensor(v).to(device) for k,v in x_train_tok.items()}
<ipython-input-54-cbedc1b8b59f>:6: UserWarning: To copy construct from a tensor,
it is recommended to use sourceTensor.clone().detach() or
sourceTensor.clone().detach().requires_grad_(True), rather than
torch.tensor(sourceTensor).
  x_test_tok = {k:torch.tensor(v).to(device) for k,v in x_test_tok.items()}
```

[55]:
```
with torch.no_grad():
  hidden_train = model(**x_train_tok)
  hidden_test = model(**x_test_tok)

# Get the [CLS] hidden states
cls_train = hidden_train.last_hidden_state[:,0,:]
cls_test = hidden_test.last_hidden_state[:,0,:]
```

[56]:
```
x_train_db = cls_train.to("cpu")
# y_train_tok

x_test_db = cls_test.to("cpu")
# y_test_tok
```

[81]:
```
lr_model2 = LogisticRegression(C=1, solver='saga', random_state=8102,␣
  ↪n_jobs=-2, max_iter=10_000)

lr_model2.fit(x_train_db,y_train_tok)

# This does not converge, with the settings used for TF-DF
# ConvergenceWarning: The max_iter was reached which means the coef_ did not␣
↪converge
# So we adjusting max_iter and experimented with C (regulation strength).

y_pred = lr_model2.predict(x_test_db)
```

[58]:
```
show_metrics(y_test_tok, y_pred, "Logistic Regression on DistilBERT hidden␣
↪states")
```

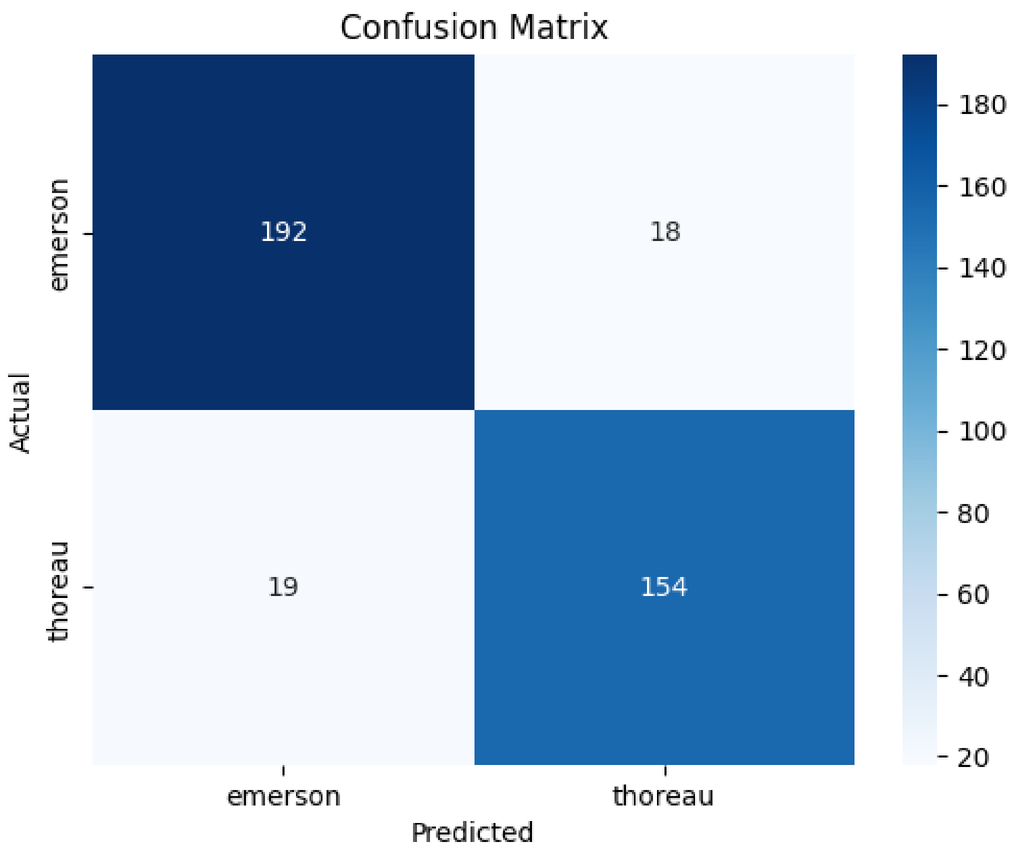
5 Logistic Regression on DistilBERT hidden states

```
              precision    recall  f1-score   support

     emerson       0.91      0.91      0.91       210
     thoreau       0.90      0.89      0.89       173

    accuracy                           0.90       383
   macro avg       0.90      0.90      0.90       383
weighted avg       0.90      0.90      0.90       383

Test accuracy: 0.9033942558746736
```



[59]:
```
rf = RandomForestClassifier()
rf.fit(x_train_db,y_train_tok)

rf.score(x_test_db,y_test_tok)

y_pred_rf = rf.predict(x_test_db)
```

[60]:
```
show_metrics(y_test_tok, y_pred_rf, "Random Forest on DistilBERT hidden states")
```

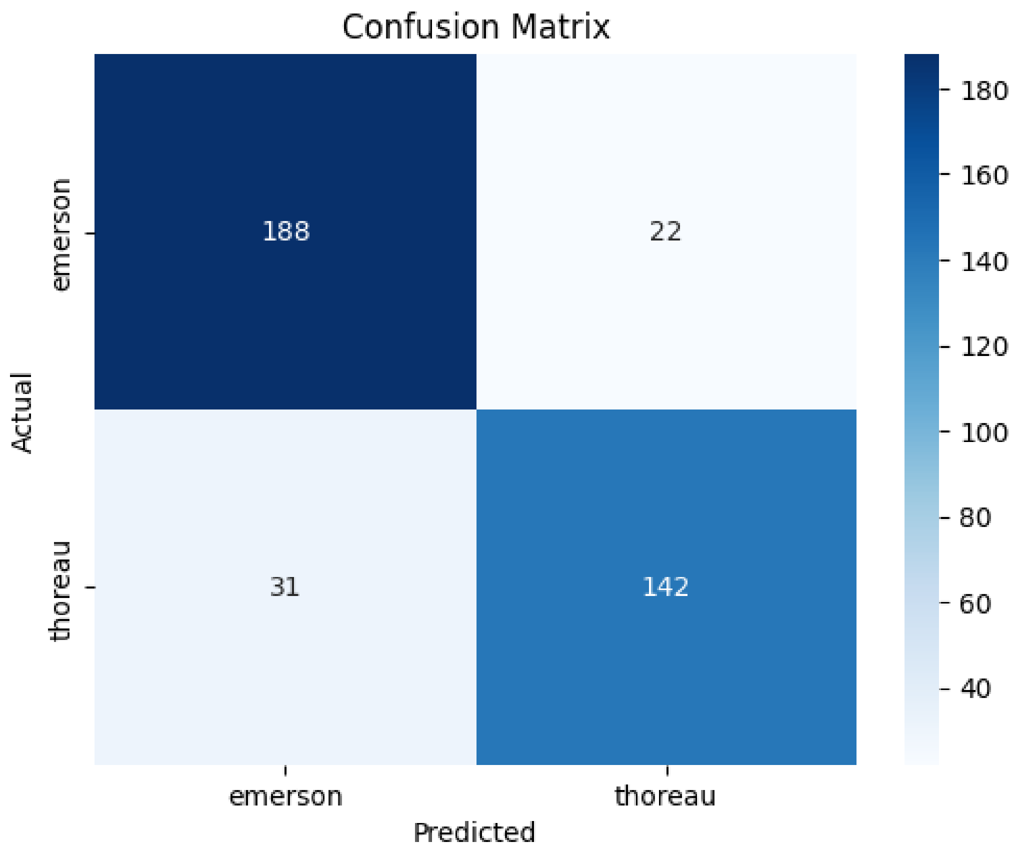
6 Random Forest on DistilBERT hidden states

```
              precision    recall  f1-score   support

     emerson       0.86      0.90      0.88       210
     thoreau       0.87      0.82      0.84       173

    accuracy                           0.86       383
   macro avg       0.86      0.86      0.86       383
weighted avg       0.86      0.86      0.86       383

Test accuracy: 0.8616187989556136
```



[61]:
```
from sklearn import svm
```

[62]:
```
# create the SVM classifier
clf = svm.SVC(kernel='rbf')

clf.fit(x_train_db,y_train_tok)

y_pred_svm = clf.predict(x_test_db)
```

[63]:
```
show_metrics(y_test_tok, y_pred_svm, "SVM on DistilBERT hidden states")
```

7 SVM on DistilBERT hidden states

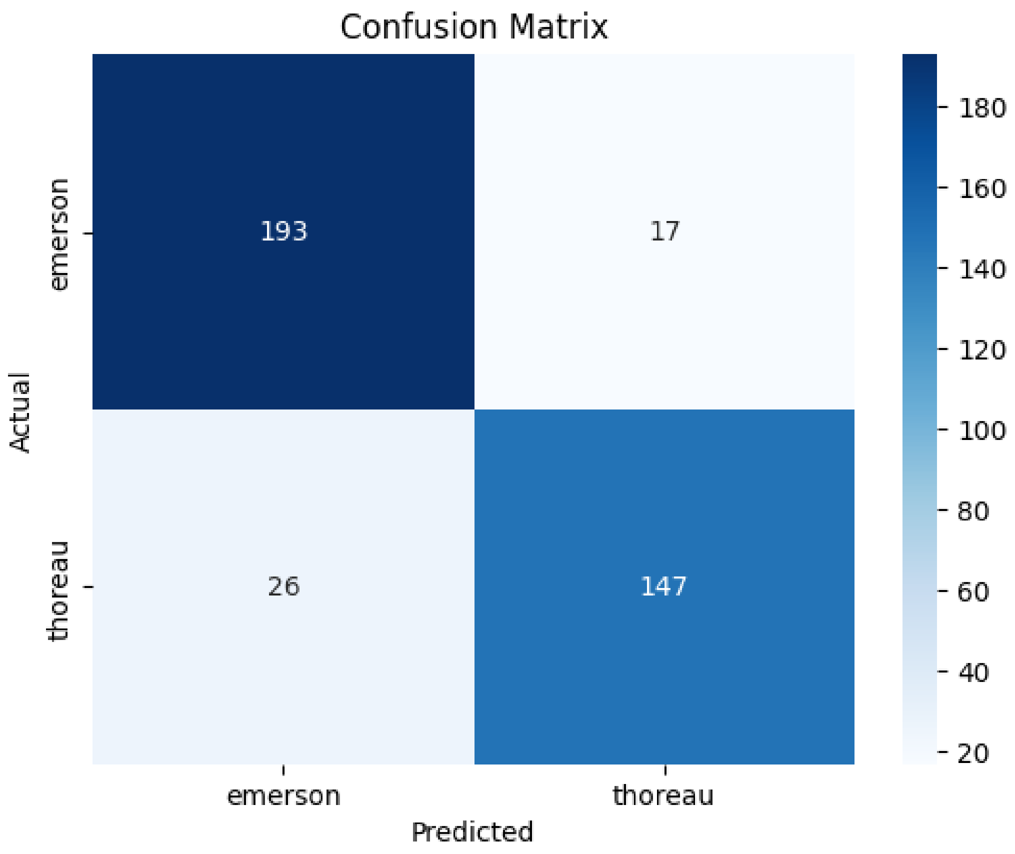
```
              precision    recall  f1-score   support

     emerson       0.88      0.92      0.90       210
     thoreau       0.90      0.85      0.87       173

    accuracy                           0.89       383
```

```
   macro avg       0.89      0.88      0.89       383
weighted avg       0.89      0.89      0.89       383

Test accuracy: 0.8877284595300261
```



[64]:
```python
from transformers import DistilBertForSequenceClassification

# Define the model with random weights, suitable for binary classification (2␣
 ↪classes)
model = DistilBertForSequenceClassification.from_pretrained(
    'distilbert-base-uncased', num_labels=2
)
# we already have the appropriate tokenizer from before.
```

```
Some weights of DistilBertForSequenceClassification were not initialized from
the model checkpoint at distilbert-base-uncased and are newly initialized:
['classifier.bias', 'classifier.weight', 'pre_classifier.bias',
'pre_classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

```
[65]: # create our optimizer
      from torch.optim import AdamW

      optimizer = AdamW(model.parameters(), lr=5e-5)
```

```
[66]: !pip install -q datasets
```

```
                              0.0/491.2 kB
? eta -:--:--
                           491.2/491.2 kB
34.0 MB/s eta 0:00:00
                              0.0/116.3

kB ? eta -:--:--
                           116.3/116.3 kB
12.8 MB/s eta 0:00:00
                              0.0/183.9

kB ? eta -:--:--
                           183.9/183.9 kB
19.4 MB/s eta 0:00:00
                              0.0/143.5

kB ? eta -:--:--
                           143.5/143.5 kB
15.5 MB/s eta 0:00:00
                              0.0/194.8

kB ? eta -:--:--
                           194.8/194.8 kB
21.4 MB/s eta 0:00:00
```

```
ERROR: pip's dependency resolver does not currently take into account
all the packages that are installed. This behaviour is the source of the
following dependency conflicts.
torch 2.6.0+cu124 requires nvidia-cublas-cu12==12.4.5.8; platform_system ==
"Linux" and platform_machine == "x86_64", but you have nvidia-cublas-cu12
12.5.3.2 which is incompatible.
torch 2.6.0+cu124 requires nvidia-cuda-cupti-cu12==12.4.127; platform_system ==
"Linux" and platform_machine == "x86_64", but you have nvidia-cuda-cupti-cu12
12.5.82 which is incompatible.
torch 2.6.0+cu124 requires nvidia-cuda-nvrtc-cu12==12.4.127; platform_system ==
"Linux" and platform_machine == "x86_64", but you have nvidia-cuda-nvrtc-cu12
12.5.82 which is incompatible.
torch 2.6.0+cu124 requires nvidia-cuda-runtime-cu12==12.4.127; platform_system
== "Linux" and platform_machine == "x86_64", but you have nvidia-cuda-runtime-
cu12 12.5.82 which is incompatible.
torch 2.6.0+cu124 requires nvidia-cudnn-cu12==9.1.0.70; platform_system ==
"Linux" and platform_machine == "x86_64", but you have nvidia-cudnn-cu12
9.3.0.75 which is incompatible.
torch 2.6.0+cu124 requires nvidia-cufft-cu12==11.2.1.3; platform_system ==
"Linux" and platform_machine == "x86_64", but you have nvidia-cufft-cu12
11.2.3.61 which is incompatible.
torch 2.6.0+cu124 requires nvidia-curand-cu12==10.3.5.147; platform_system ==
"Linux" and platform_machine == "x86_64", but you have nvidia-curand-cu12
10.3.6.82 which is incompatible.
torch 2.6.0+cu124 requires nvidia-cusolver-cu12==11.6.1.9; platform_system ==
"Linux" and platform_machine == "x86_64", but you have nvidia-cusolver-cu12
11.6.3.83 which is incompatible.
torch 2.6.0+cu124 requires nvidia-cusparse-cu12==12.3.1.170; platform_system ==
"Linux" and platform_machine == "x86_64", but you have nvidia-cusparse-cu12
12.5.1.3 which is incompatible.
torch 2.6.0+cu124 requires nvidia-nvjitlink-cu12==12.4.127; platform_system ==
"Linux" and platform_machine == "x86_64", but you have nvidia-nvjitlink-cu12
12.5.82 which is incompatible.
gcsfs 2025.3.2 requires fsspec==2025.3.2, but you have fsspec 2024.12.0 which is
incompatible.
```

[67]:
```
from sklearn import preprocessing

# Create a copy of our dataframe
trans_df = combined_df.copy()

# drop the preprocessed text column which we aren't using.
trans_df.drop("final_text", axis=1, inplace=True)

# transform our labels into numeric values.
le = preprocessing.LabelEncoder()
my_labels = trans_df["label"].tolist()
le.fit(my_labels)

my_cat_labels = le.classes_
trans_df["label"] = le.transform(trans_df["label"])

print(f"{my_cat_labels=}")

trans_df.info()
trans_df.describe()
trans_df.head()
```

```
my_cat_labels=array(['emerson', 'thoreau'], dtype='<U7')
<class 'pandas.core.frame.DataFrame'>
Index: 1911 entries, 153 to 77
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   text    1911 non-null   object
 1   label   1911 non-null   int64
dtypes: int64(1), object(1)
memory usage: 44.8+ KB
```

[67]:
```
                                                  text  label
153  During this fast they abstain from the\n\ngrat…      1
638  The life of man is a self-evolving circle,[696…      0
66   But perhaps a man is not required to bury hims…      1
409  Fine manners[398] show\n\nthemselves formidabl…      0
201  We must learn to reawaken and keep ourselves a…      1
```

[68]:
```
from datasets import Dataset
from transformers import AutoTokenizer

# for simplicity, we are just splitting the dataset again.
train_df, test_df = train_test_split(trans_df, test_size=0.2, random_state=4909)
```

```
train_dataset = Dataset.from_pandas(train_df)
test_dataset = Dataset.from_pandas(test_df)

def tokenize_data(examples):
    return tokenizer(examples["text"], truncation=True)

tokenized_train = train_dataset.map(tokenize_data, batched=True)
tokenized_test = test_dataset.map(tokenize_data, batched=True)
```

```
Map:   0%|          | 0/1528 [00:00<?, ? examples/s]

Map:   0%|          | 0/383 [00:00<?, ? examples/s]
```

[69]:
```
from transformers import Trainer, TrainingArguments, DataCollatorWithPadding

data_collator = DataCollatorWithPadding(tokenizer=tokenizer)

training_args = TrainingArguments(
    output_dir="./results",
    learning_rate=2e-4,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=5,
    weight_decay=0.01,
    evaluation_strategy="epoch",
    logging_strategy="epoch"
)

# Define Trainer object for training the model
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_train,
    eval_dataset=tokenized_test,
    tokenizer=tokenizer,
    data_collator=data_collator,
)

trainer.train()
```

```
/usr/local/lib/python3.11/dist-packages/transformers/training_args.py:1611:
FutureWarning: `evaluation_strategy` is deprecated and will be removed in
version 4.46 of  Transformers. Use `eval_strategy` instead
  warnings.warn(
<ipython-input-69-c2f16bbd5fc2>:17: FutureWarning: `tokenizer` is deprecated and
will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class`
instead.
```

```
  trainer = Trainer(
wandb: WARNING The `run_name` is currently set to the same
value as `TrainingArguments.output_dir`. If this was not intended, please
specify a different run name by setting the `TrainingArguments.run_name`
parameter.
wandb: Using wandb-core as the SDK backend.  Please refer to
https://wandb.me/wandb-core for more information.
```

```
<IPython.core.display.Javascript object>
```

```
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server
locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here:
https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter:
```

```
 ··········
```

```
wandb: WARNING If you're specifying your api key in code,
ensure this code is not shared publicly.
wandb: WARNING Consider setting the WANDB_API_KEY
environment variable, or running `wandb login` from the command line.
wandb: No netrc file found, creating one.
wandb: Appending key for api.wandb.ai to your netrc file:
/root/.netrc
wandb: Currently logged in as: ranton (ranton-
fieldjay-com) to https://api.wandb.ai. Use `wandb login
--relogin` to force relogin
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>
```

[69]:
```
TrainOutput(global_step=955, training_loss=0.1480980508614585,
metrics={'train_runtime': 112.7924, 'train_samples_per_second': 67.735,
'train_steps_per_second': 8.467, 'total_flos': 434145340670208.0, 'train_loss':
0.1480980508614585, 'epoch': 5.0})
```

[70]:
```
# Save the model.
trainer.save_model('model')
```

[71]:
```
# This is how you can load the model.
# from transformers import AutoModelForSequenceClassification
# model = AutoModelForSequenceClassification.from_pretrained("./model")
```

[72]:
```
def predictor(text):
  #inputs = tokenizer(text, return_tensors="pt")
  inputs = tokenizer(text, padding=True, truncation=True, return_tensors="pt")
  inputs = {k:torch.tensor(v).to(device) for k,v in inputs.items()}

  with torch.no_grad():
      logits = model(**inputs).logits
  predictions = torch.argmax(logits, dim=-1)
  return predictions
```

[73]:
```
x_test_trans = test_dataset["text"]
y_test_trans = test_dataset["label"]

# sanity test a few inference inputs.
for txt, lbl in zip(x_test_trans[:5], y_test_trans[:5]):
  pred = predictor( txt)
  print(f"{my_cat_labels[lbl]}: pred={my_cat_labels[pred]}, {txt=}")
```

```
<ipython-input-72-118abc29aba2>:4: UserWarning: To copy construct from a tensor,
it is recommended to use sourceTensor.clone().detach() or
sourceTensor.clone().detach().requires_grad_(True), rather than
torch.tensor(sourceTensor).
  inputs = {k:torch.tensor(v).to(device) for k,v in inputs.items()}
```

```
thoreau: pred=thoreau, txt='He should have\n\ngone up garret at once. "What!"
exclaim a million Irishmen starting up\n\nfrom all the shanties in the land, "is
not this railroad which we have\n\nbuilt a good thing?" Yes, I answer,
_comparatively_ good, that is, you\n\nmight have done worse; but I wish, as you
are brothers of mine, that\n\nyou could have spent your time better than digging
in this dirt.\n\n\n\n\n\n\n\n'
thoreau: pred=thoreau, txt='The exact cost of my house, paying the usual
price\n\nfor such materials as I used, but not counting the work, all of
which\n\nwas done by myself, was as follows; and I give the details because
very\n\nfew are able to tell exactly what their houses cost, and fewer
still,\n\nif any, the separate cost of the various materials which compose
them:-\n\n\n\n\n\n    Boards… $ 8.03½, mostly shanty
boards.\n\n     Refuse shingles for roof sides,..  4.00\n\n
Laths,…  1.25\n\n    Two second-hand windows\n\n
with glass,…  2.43\n\n    One thousand old brick,…
4.00\n\n    Two casks of lime,…  2.40  That was high.\n\n
Hair,…  0.31  More than I needed.\n\n     Mantle-tree
iron,…  0.15\n\n    Nails,…  3.90\n\n
Hinges and screws,…  0.14\n\n    Latch,…
0.10\n\n    Chalk,…  0.01\n\n
Transportation,…  1.40   I carried a good part\n\n
```

```
---- on my back.\n\n         In all,…
$28.12½\n\n\n\n\n\nThese are all the materials excepting the timber stones and
sand, which\n\nI claimed by squatter’s right.'
emerson: pred=emerson, txt='[Footnote 181: _The other terror. _ The first,
conformity, has just\n\nbeen treated.]\n\n\n\n [Footnote 182: Consistency.'
thoreau: pred=thoreau, txt='It is the luxurious and dissipated who set the
fashions which the herd\n\nso diligently follow. The traveller who stops at the
best houses, so\n\ncalled, soon discovers this, for the publicans presume him to
be a\n\nSardanapalus, and if he resigned himself to their tender mercies
he\n\nwould soon be completely emasculated. I think that in the railroad
car\n\nwe are inclined to spend more on luxury than on safety and
convenience,\n\nand it threatens without attaining these to become no better
than a\n\nmodern drawing room, with its divans, and ottomans, and sun-shades,
and\n\na hundred other oriental things, which we are taking west with
us,\n\ninvented for the ladies of the harem and the effeminate natives of
the\n\nCelestial Empire, which Jonathan should be ashamed to know the
names\n\nof.'
thoreau: pred=thoreau, txt='So the Muse fables. But therein, as I found, dwelt
now John Field, an\n\nIrishman, and his wife, and several children, from the
broad-faced boy\n\nwho assisted his father at his work, and now came running by
his side\n\nfrom the bog to escape the rain, to the wrinkled, sibyl-
like,\n\ncone-headed infant that sat upon its father’s knee as in the palaces
of\n\nnobles, and looked out from its home in the midst of wet and
hunger\n\ninquisitively upon the stranger, with the privilege of infancy,
not\n\nknowing but it was the last of a noble line, and the hope and
cynosure\n\nof the world, instead of John Field’s poor starveling brat. There
we\n\nsat together under that part of the roof which leaked the least,
while\n\nit showered and thundered without.'
```

[74]:
```python
y_pred_trans = [predictor(txt) for txt in x_test_trans]

for txt, lbl, pred in zip(x_test_trans[:5], y_test_trans[:5], y_pred_trans[:5]):
  print(f"{my_cat_labels[lbl]}: pred={my_cat_labels[pred]}, {txt=}")
```

```
<ipython-input-72-118abc29aba2>:4: UserWarning: To copy construct from a tensor,
it is recommended to use sourceTensor.clone().detach() or
sourceTensor.clone().detach().requires_grad_(True), rather than
torch.tensor(sourceTensor).
  inputs = {k:torch.tensor(v).to(device) for k,v in inputs.items()}
```

```
thoreau: pred=thoreau, txt='He should have\n\ngone up garret at once. "What!"
exclaim a million Irishmen starting up\n\nfrom all the shanties in the land, "is
not this railroad which we have\n\nbuilt a good thing?" Yes, I answer,
_comparatively_ good, that is, you\n\nmight have done worse; but I wish, as you
are brothers of mine, that\n\nyou could have spent your time better than digging
in this dirt.\n\n\n\n\n\n\n\n'
thoreau: pred=thoreau, txt='The exact cost of my house, paying the usual
price\n\nfor such materials as I used, but not counting the work, all of
```

```
which\n\nwas done by myself, was as follows; and I give the details because
very\n\nfew are able to tell exactly what their houses cost, and fewer
still,\n\nif any, the separate cost of the various materials which compose
them:-\n\n\n\n\n\n    Boards… $ 8.03½, mostly shanty
boards.\n\n     Refuse shingles for roof sides,..  4.00\n\n
Laths,…  1.25\n\n    Two second-hand windows\n\n
with glass,…  2.43\n\n    One thousand old brick,…
4.00\n\n    Two casks of lime,…  2.40  That was high.\n\n
Hair,…  0.31  More than I needed.\n\n     Mantle-tree
iron,…  0.15\n\n    Nails,…  3.90\n\n
Hinges and screws,…  0.14\n\n    Latch,…
0.10\n\n    Chalk,…  0.01\n\n
Transportation,…  1.40   I carried a good part\n\n
---- on my back.\n\n         In all,…
$28.12½\n\n\n\n\n\nThese are all the materials excepting the timber stones and
sand, which\n\nI claimed by squatter's right.'
emerson: pred=emerson, txt='[Footnote 181: _The other terror. _ The first,
conformity, has just\n\nbeen treated.]\n\n\n\n [Footnote 182: Consistency.'
thoreau: pred=thoreau, txt='It is the luxurious and dissipated who set the
fashions which the herd\n\nso diligently follow. The traveller who stops at the
best houses, so\n\ncalled, soon discovers this, for the publicans presume him to
be a\n\nSardanapalus, and if he resigned himself to their tender mercies
he\n\nwould soon be completely emasculated. I think that in the railroad
car\n\nwe are inclined to spend more on luxury than on safety and
convenience,\n\nand it threatens without attaining these to become no better
than a\n\nmodern drawing room, with its divans, and ottomans, and sun-shades,
and\n\na hundred other oriental things, which we are taking west with
us,\n\ninvented for the ladies of the harem and the effeminate natives of
the\n\nCelestial Empire, which Jonathan should be ashamed to know the
names\n\nof.'
thoreau: pred=thoreau, txt='So the Muse fables. But therein, as I found, dwelt
now John Field, an\n\nIrishman, and his wife, and several children, from the
broad-faced boy\n\nwho assisted his father at his work, and now came running by
his side\n\nfrom the bog to escape the rain, to the wrinkled, sibyl-
like,\n\ncone-headed infant that sat upon its father's knee as in the palaces
of\n\nnobles, and looked out from its home in the midst of wet and
hunger\n\ninquisitively upon the stranger, with the privilege of infancy,
not\n\nknowing but it was the last of a noble line, and the hope and
cynosure\n\nof the world, instead of John Field's poor starveling brat. There
we\n\nsat together under that part of the roof which leaked the least,
while\n\nit showered and thundered without.'
```

```
[75]: y_pred_trans = [torch.tensor(v).cpu() for v in y_pred_trans]
```

```
<ipython-input-75-29869180259a>:1: UserWarning: To copy construct from a tensor,
it is recommended to use sourceTensor.clone().detach() or
sourceTensor.clone().detach().requires_grad_(True), rather than
torch.tensor(sourceTensor).
```

```
y_pred_trans = [torch.tensor(v).cpu() for v in y_pred_trans]
```

[76]:
```
y_test_trans = [torch.tensor(v).cpu() for v in y_test_trans]
```

[77]:
```
show_metrics(y_pred_trans, y_test_trans, "Fine-tuned DistilBERT")
```

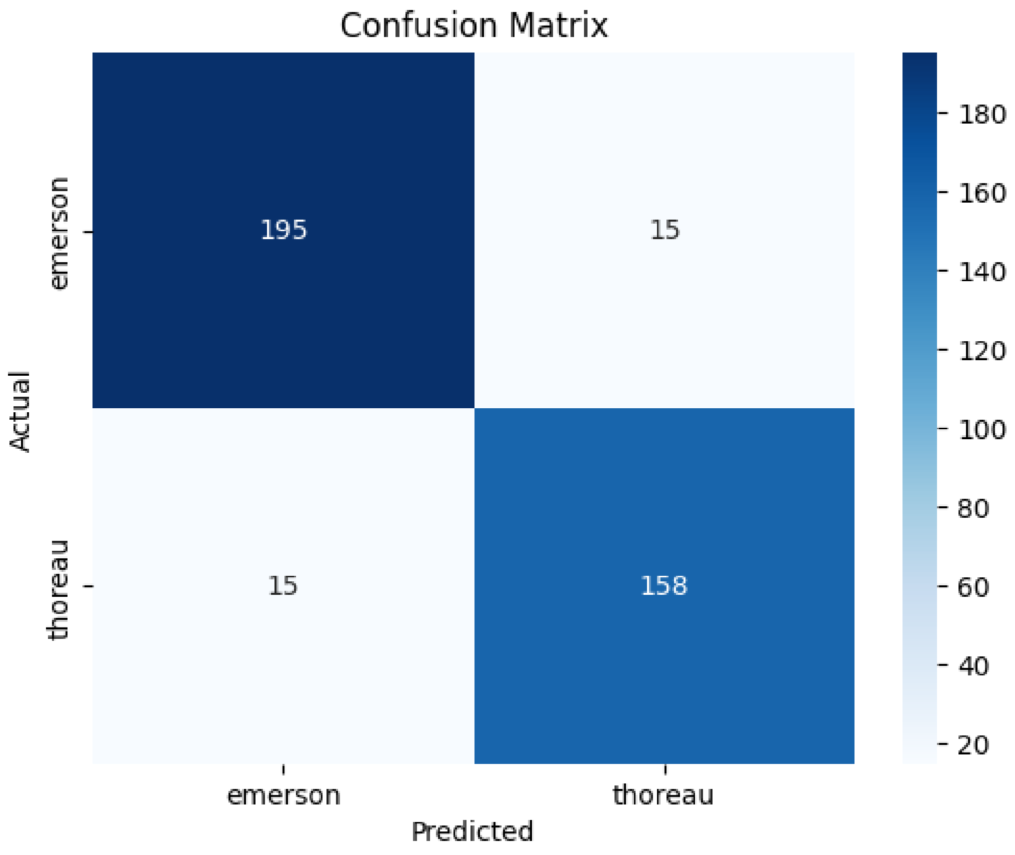
8 Fine-tuned DistilBERT

```
              precision    recall  f1-score   support

           0       0.93      0.93      0.93       210
           1       0.91      0.91      0.91       173

    accuracy                           0.92       383
   macro avg       0.92      0.92      0.92       383
weighted avg       0.92      0.92      0.92       383

Test accuracy: 0.9216710182767625
```



8.1 Let's check out the test samples that are misclassified.

[78]:
```
def scalar_from_tensor(t):
  if t.dim() == 0:
    return t.item()
  elif t.dim() == 1:
    return t[0].item()
  else:
    raise ValueError(f"Unexpected tensor dimension: {t.dim()}")
```

[79]:
```
y_test_trans = [scalar_from_tensor(t) for t in y_test_trans]
y_pred_trans = [scalar_from_tensor(t) for t in y_pred_trans]

print(f"y_test_trans: {y_test_trans[:5]}")
print(f"y_pred_trans: {y_pred_trans[:5]}")
```

```
y_test_trans: [1, 1, 0, 1, 1]
y_pred_trans: [1, 1, 0, 1, 1]
```

[80]:
```
print("my_cat_labels")

rows = []
for i, (txt, lbl, pred) in enumerate(zip(x_test_trans, y_test_trans,␣
 ↪y_pred_trans)):
  if lbl != pred:
    print(f"{lbl=},{pred=}")
    row =(my_cat_labels[lbl], my_cat_labels[pred], txt)
    print(f"{row=}")
    rows.append(row)

n_miss = len(rows)
print(f"Count of misclassified = {n_miss}")
misclassified_df = pd.DataFrame(rows, columns=["actual", "predicted", "text"])
misclassified_df.head(n_miss)
```

```
my_cat_labels
lbl=1,pred=0
row=(np.str_('thoreau'), np.str_('emerson'), '1.F.6. INDEMNITY - You agree to
indemnify and hold the Foundation, the\n\ntrademark owner, any agent or employee
of the Foundation, anyone\n\nproviding copies of Project Gutenberg  electronic
works in\n\naccordance with this agreement, and any volunteers associated with
the\n\nproduction, promotion and distribution of Project
Gutenberg \n\nelectronic works, harmless from all liability, costs and
expenses,\n\nincluding legal fees, that arise directly or indirectly from any
of\n\nthe following which you do or cause to occur: (a) distribution of
this\n\nor any Project Gutenberg  work, (b) alteration, modification,
or\n\nadditions or deletions to any Project Gutenberg  work, and (c)
any\n\nDefect you cause.\n\n\n\n Section 2.')
```

```
lbl=0,pred=1
row=(np.str_('emerson'), np.str_('thoreau'), 'The city is recruited from the
country. In the year\n\n1805, it is said, every legitimate monarch in Europe was
imbecile. The\n\ncity would have died out, rotted, and exploded, long ago, but
that it\n\nwas reinforced from the fields.')
lbl=1,pred=0
row=(np.str_('thoreau'), np.str_('emerson'), 'The\n\nsuccess of great scholars
and thinkers is commonly a courtier-like\n\nsuccess, not kingly, not manly. They
make shift to live merely by\n\nconformity, practically as their fathers did,
and are in no sense the\n\nprogenitors of a nobler race of men. But why do men
degenerate ever?\n\n')
lbl=0,pred=1
row=(np.str_('emerson'), np.str_('thoreau'), 'If you discover a\n\ndefect in
this electronic work within 90 days of receiving it, you can\n\nreceive a refund
of the money (if any) you paid for it by sending a\n\nwritten explanation to the
person you received the work from. If you\n\nreceived the work on a physical
medium, you must return the medium\n\nwith your written explanation. The person
or entity that provided you\n\nwith the defective work may elect to provide a
replacement copy in\n\nlieu of a refund.')
lbl=0,pred=1
row=(np.str_('emerson'), np.str_('thoreau'), 'Do not charge a fee for access to,
viewing, displaying,\n\nperforming, copying or distributing any Project
Gutenberg  works\n\nunless you comply with paragraph 1.E.8 or
1.E.9.\n\n\n\n1.E.8. You may charge a reasonable fee for copies of or
providing\n\naccess to or distributing Project Gutenberg  electronic
works\n\nprovided that:\n\n\n\n    • You pay a royalty fee of 20% of the gross
profits you derive from\n\n        the use of Project Gutenberg  works
calculated using the method\n\n        you already use to calculate your
applicable taxes. The fee is owed\n\n        to the owner of the Project
Gutenberg  trademark, but he has\n\n        agreed to donate royalties under
this paragraph to the Project\n\n        Gutenberg Literary Archive Foundation.
Royalty payments must be paid\n\n        within 60 days following each date on
which you prepare (or are\n\n        legally required to prepare) your periodic
tax returns.')
lbl=1,pred=0
row=(np.str_('thoreau'), np.str_('emerson'), 'Comparatively, he is always
strong, original, and, above all,\n\npractical. Still his quality is not wisdom,
but prudence. The lawyer's\n\ntruth is not Truth, but consistency or a
consistent expediency.')
lbl=0,pred=1
row=(np.str_('emerson'), np.str_('thoreau'), 'It\n\nexists because of the
efforts of hundreds of volunteers and donations\n\nfrom people in all walks of
life.\n\n\n\n Volunteers and financial support to provide volunteers with
the\n\nassistance they need are critical to reaching Project
Gutenberg 's\n\ngoals and ensuring that the Project Gutenberg  collection
will\n\nremain freely available for generations to come. In 2001, the
Project\n\nGutenberg Literary Archive Foundation was created to provide a
secure\n\nand permanent future for Project Gutenberg  and future\n\ngenerations.
```

```
To learn more about the Project Gutenberg Literary\n\nArchive Foundation and how
your efforts and donations can help, see\n\nSections 3 and 4 and the Foundation
information page at www.gutenberg.org.\n\n\n\n Section 3.')
lbl=0,pred=1
row=(np.str_('emerson'), np.str_('thoreau'), 'and what a compensation for
the\n\nshortness of life! All is done to his hand. The world has brought
him\n\nthus far on his way. The human race has gone out before him, sunk
the\n\nhills, filled the hollows, and bridged the rivers.')
lbl=1,pred=0
row=(np.str_('thoreau'), np.str_('emerson'), 'How much\n\nmore admirable the
Bhagvat-Geeta than all the ruins of the East! Towers\n\nand temples are the
luxury of princes. A simple and independent mind\n\ndoes not toil at the bidding
of any prince.')
lbl=0,pred=1
row=(np.str_('emerson'), np.str_('thoreau'), 'You may use this eBook for nearly
any purpose such as creation\n\nof derivative works, reports, performances and
research. Project\n\nGutenberg eBooks may be modified and printed and given
away-you may\n\ndo practically ANYTHING in the United States with eBooks not
protected\n\nby U.S. copyright law. Redistribution is subject to the
trademark\n\nlicense, especially commercial redistribution.\n\n\n\n\n\n')
lbl=0,pred=1
row=(np.str_('emerson'), np.str_('thoreau'), 'See the\n\ninvestment of capital
in aqueducts, made useless by hydraulics;\n\nfortifications, by gunpowder; roads
and canals, by railways; sails, by\n\nsteam; steam, by electricity.\n\n\n\n You
admire this tower of granite, weathering the hurts of so many\n\nages. Yet a
little waving hand built this huge wall, and that which\n\nbuilds is better than
that which is built.')
lbl=0,pred=1
row=(np.str_('emerson'), np.str_('thoreau'), 'The length of the discourse
indicates the\n\ndistance of thought betwixt the speaker and the hearer. If they
were\n\nat a perfect understanding in any part, no words would be
necessary\n\nthereon. If at one in all parts, no words would be
suffered.\n\n\n\n')
lbl=0,pred=1
row=(np.str_('emerson'), np.str_('thoreau'), 'All the\n\nmass has been treated,
with more or less skill, by every playwright,\n\nand the prompter has the soiled
and tattered manuscripts. It is now no\n\nlonger possible to say who wrote them
first. They have been the\n\nproperty of the Theater so long, and so many rising
geniuses have\n\nenlarged or altered them, inserting a speech, or a whole scene,
or\n\nadding a song, that no man can any longer claim copyright in this
work\n\nof numbers. Happily, no man wishes to.')
lbl=0,pred=1
row=(np.str_('emerson'), np.str_('thoreau'), 'Of course, we hope\n\nthat you
will support the Project Gutenberg mission of promoting\n\nfree access to
electronic works by freely sharing Project Gutenberg \n\nworks in compliance
with the terms of this agreement for keeping the\n\nProject Gutenberg name
associated with the work. You can easily\n\ncomply with the terms of this
agreement by keeping this work in the\n\nsame format with its attached full
```

```
Project Gutenberg License when\n\nyou share it without charge with
others.\n\n\n\n 1.D. The copyright laws of the place where you are located also
govern\n\nwhat you can do with this work. Copyright laws in most countries
are\n\nin a constant state of change.')
lbl=1,pred=0
row=(np.str_('thoreau'), np.str_('emerson'), 'In earlier ages, in some
countries, every function was reverently\n\nspoken of and regulated by law.
Nothing was too trivial for the Hindoo\n\nlawgiver, however offensive it may be
to modern taste. He teaches how\n\nto eat, drink, cohabit, void excrement and
urine, and the like,\n\nelevating what is mean, and does not falsely excuse
himself by calling\n\nthese things trifles.\n\n\n\n Every man is the builder of
a temple, called his body, to the god he\n\nworships, after a style purely his
own, nor can he get off by hammering\n\nmarble instead.')
lbl=1,pred=0
row=(np.str_('thoreau'), np.str_('emerson'), 'how much more important to know
what that is which was never\n\nold! "Kieou-he-yu (great dignitary of the state
of Wei) sent a man to\n\nKhoung-tseu to know his news. Khoung-tseu caused the
messenger to be\n\nseated near him, and questioned him in these terms: What is
your master\n\ndoing? The messenger answered with respect: My master desires
to\n\ndiminish the number of his faults, but he cannot come to the end
of\n\nthem.')
lbl=0,pred=1
row=(np.str_('emerson'), np.str_('thoreau'), '1.F.\n\n\n\n1.F.1. Project
Gutenberg volunteers and employees expend considerable\n\neffort to identify, do
copyright research on, transcribe and proofread\n\nworks not protected by U.S.
copyright law in creating the Project\n\nGutenberg collection. Despite these
efforts, Project Gutenberg \n\nelectronic works, and the medium on which they
may be stored, may\n\ncontain "Defects," such as, but not limited to,
incomplete, inaccurate\n\nor corrupt data, transcription errors, a copyright or
other\n\nintellectual property infringement, a defective or damaged disk
or\n\nother medium, a computer virus, or computer codes that damage or\n\ncannot
be read by your equipment.\n\n\n\n1.F.2. LIMITED WARRANTY, DISCLAIMER OF DAMAGES
-')
lbl=1,pred=0
row=(np.str_('thoreau'), np.str_('emerson'), 'Our whole life is startlingly
moral. There is never an instant’s truce\n\nbetween virtue and vice. Goodness is
the only investment that never\n\nfails. In the music of the harp which trembles
round the world it is\n\nthe insisting on this which thrills us.')
lbl=1,pred=0
row=(np.str_('thoreau'), np.str_('emerson'), 'Nature puts no question
and\n\nanswers none which we mortals ask. She has long ago taken
her\n\nresolution. "O Prince, our eyes contemplate with admiration
and\n\ntransmit to the soul the wonderful and varied spectacle of
this\n\nuniverse. The night veils without doubt a part of this
glorious\n\ncreation; but day comes to reveal to us this great work, which
extends\n\nfrom earth even into the plains of the ether."\n\n\n\n Then to my
morning work.')
lbl=0,pred=1
```

```
row=(np.str_('emerson'), np.str_('thoreau'), 'Better than the hand and nimbler
was the\n\ninvisible thought which wrought through it; and thus ever, behind
the\n\ncoarse effect, is a fine cause, which, being narrowly seen, is
itself\n\nthe effect of a finer cause. Everything looks permanent until
its\n\nsecret is known. A rich estate appears to women and children a
firm\n\nand lasting fact; to a merchant, one easily created out of
any\n\nmaterials, and easily lost. An orchard, good tillage, good
grounds,\n\nseem a fixture, like a gold mine, or a river, to a citizen; but to
a\n\nlarge farmer, not much more fixed than the state of the crop.')
lbl=1,pred=0
row=(np.str_('thoreau'), np.str_('emerson'), '1.E. Unless you have removed all
references to Project Gutenberg:\n\n\n\n1.E.1. The following sentence, with
active links to, or other\n\nimmediate access to, the full Project Gutenberg
License must appear\n\nprominently whenever any copy of a Project Gutenberg
work (any work\n\non which the phrase "Project Gutenberg" appears, or with which
the\n\nphrase "Project Gutenberg" is associated) is accessed,
displayed,\n\nperformed, viewed, copied or distributed:\n\n\n\n    This eBook is
for the use of anyone anywhere in the United States and most\n\n    other parts
of the world at no cost and with almost no restrictions\n\n    whatsoever. You
may copy it, give it away or re-use it under the terms\n\n    of the Project
Gutenberg License included with this eBook or online\n\n    at
www.gutenberg.org. If you\n\n    are not located in the United States, you will
have to check the laws\n\n    of the country where you are located before using
this eBook.\n\n  \n\n1.E.2.')
lbl=0,pred=1
row=(np.str_('emerson'), np.str_('thoreau'), 'You provide, in accordance with
paragraph 1.F.3, a full refund of\n\n        any money paid for a work or a
replacement copy, if a defect in the\n\n        electronic work is discovered
and reported to you within 90 days of\n\n        receipt of the work.\n\n
\n\n     • You comply with all other terms of this agreement for free\n\n
distribution of Project Gutenberg works.\n\n    \n\n\n\n1.E.9. If you wish to
charge a fee or distribute a Project\n\nGutenberg electronic work or group of
works on different terms than\n\nare set forth in this agreement, you must
obtain permission in writing\n\nfrom the Project Gutenberg Literary Archive
Foundation, the manager of\n\nthe Project Gutenberg trademark.')
lbl=1,pred=0
row=(np.str_('thoreau'), np.str_('emerson'), '1.D. The copyright laws of the
place where you are located also govern\n\nwhat you can do with this work.
Copyright laws in most countries are\n\nin a constant state of change. If you
are outside the United States,\n\ncheck the laws of your country in addition to
the terms of this\n\nagreement before downloading, copying, displaying,
performing,\n\ndistributing or creating derivative works based on this work or
any\n\nother Project Gutenberg work.')
lbl=1,pred=0
row=(np.str_('thoreau'), np.str_('emerson'), 'I will just try these three
sentences of Con-fut-see; they may fetch\n\nthat state about again. I know not
whether it was the dumps or a\n\nbudding ecstasy. Mem. There never is but one
opportunity of a kind.\n\n\n\n_Poet.')
```

```
lbl=0,pred=1
row=(np.str_('emerson'), np.str_('thoreau'), 'It may only be\n\nused on or
associated in any way with an electronic work by people who\n\nagree to be bound
by the terms of this agreement. There are a few\n\nthings that you can do with
most Project Gutenberg electronic works\n\neven without complying with the full
terms of this agreement. See\n\nparagraph 1.C below. There are a lot of things
you can do with Project\n\nGutenberg electronic works if you follow the terms
of this\n\nagreement and help preserve free future access to Project
Gutenberg \n\nelectronic works.')
lbl=1,pred=0
row=(np.str_('thoreau'), np.str_('emerson'), 'The Foundation's EIN or federal
tax identification\n\nnumber is 64-6221541. Contributions to the Project
Gutenberg Literary\n\nArchive Foundation are tax deductible to the full extent
permitted by\n\nU.S. federal laws and your state's laws.\n\n\n\n The
Foundation's business office is located at 809 North 1500 West,\n\nSalt Lake
City, UT 84116, (801) 596-1887.')
lbl=1,pred=0
row=(np.str_('thoreau'), np.str_('emerson'), 'Poetry and art, and the fairest
and most memorable\n\nof the actions of men, date from such an hour. All poets
and heroes,\n\nlike Memnon, are the children of Aurora, and emit their music
at\n\nsunrise. To him whose elastic and vigorous thought keeps pace with
the\n\nsun, the day is a perpetual morning. It matters not what the clocks
say\n\nor the attitudes and labors of men.')
lbl=1,pred=0
row=(np.str_('thoreau'), np.str_('emerson'), 'No wonder that the\n\nearth
expresses itself outwardly in leaves, it so labors with the idea\n\ninwardly.
The atoms have already learned this law, and are pregnant by\n\nit. The
overhanging leaf sees here its prototype.')
lbl=1,pred=0
row=(np.str_('thoreau'), np.str_('emerson'), 'Alas! how little does the memory
of these human inhabitants\n\nenhance the beauty of the landscape! Again,
perhaps, Nature will try,\n\nwith me for a first settler, and my house raised
last spring to be the\n\noldest in the hamlet.\n\n\n\n I am not aware that any
man has ever built on the spot which I occupy.\n\n Deliver me from a city built
on the site of a more ancient city, whose\n\nmaterials are ruins, whose gardens
cemeteries.')
lbl=0,pred=1
row=(np.str_('emerson'), np.str_('thoreau'), 'Has\n\nhe talents? has he
enterprises? has he knowledge? It boots not.\n\n Infinitely alluring and
attractive was he to you yesterday, a great\n\nhope, a sea to swim in; now, you
have found his shores, found it a\n\npond, and you care not if you never see it
again.\n\n\n\n')
Count of misclassified = 30
```

[80]:

```
     actual predicted                                               text
0   thoreau   emerson  1.F.6. INDEMNITY - You agree to indemnify and …
1   emerson   thoreau  The city is recruited from the country. In the…
```

```
2   thoreau   emerson  The\n\nsuccess of great scholars and thinkers …
3   emerson   thoreau  If you discover a\n\ndefect in this electronic…
4   emerson   thoreau  Do not charge a fee for access to, viewing, di…
5   thoreau   emerson  Comparatively, he is always strong, original, …
6   emerson   thoreau  It\n\nexists because of the efforts of hundred…
7   emerson   thoreau  and what a compensation for the\n\nshortness o…
8   thoreau   emerson  How much\n\nmore admirable the Bhagvat-Geeta t…
9   emerson   thoreau  You may use this eBook for nearly any purpose …
10  emerson   thoreau  See the\n\ninvestment of capital in aqueducts,…
11  emerson   thoreau  The length of the discourse indicates the\n\nd…
12  emerson   thoreau  All the\n\nmass has been treated, with more or…
13  emerson   thoreau  Of course, we hope\n\nthat you will support th…
14  thoreau   emerson  In earlier ages, in some countries, every func…
15  thoreau   emerson  how much more important to know what that is w…
16  emerson   thoreau  1.F.\n\n\n\n1.F.1. Project Gutenberg volunteer…
17  thoreau   emerson  Our whole life is startlingly moral. There is …
18  thoreau   emerson  Nature puts no question and\n\nanswers none wh…
19  emerson   thoreau  Better than the hand and nimbler was the\n\nin…
20  thoreau   emerson  1.E. Unless you have removed all references to…
21  emerson   thoreau  You provide, in accordance with paragraph 1.F…
22  thoreau   emerson  1.D. The copyright laws of the place where you…
23  thoreau   emerson  I will just try these three sentences of Con-f…
24  emerson   thoreau  It may only be\n\nused on or associated in any…
25  thoreau   emerson  The Foundation's EIN or federal tax identifica…
26  thoreau   emerson  Poetry and art, and the fairest and most memor…
27  thoreau   emerson  No wonder that the\n\nearth expresses itself o…
28  thoreau   emerson  Alas! how little does the memory of these huma…
29  emerson   thoreau  Has\n\nhe talents? has he enterprises? has he …
```

[80]: