Praful_file2_h2

February 15, 2024

Spam Detection HW

Read complete instructions before starting the HW

1 Installing/Importing Modules

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

[2]:
```
!python -m spacy download en_core_web_sm
```

```
Collecting en-core-web-sm==3.7.1
  Downloading https://github.com/explosion/spacy-
models/releases/download/en_core_web_sm-3.7.1/en_core_web_sm-3.7.1-py3-none-
any.whl (12.8 MB)
                              12.8/12.8 MB
24.2 MB/s eta 0:00:00
Requirement already satisfied: spacy<3.8.0,>=3.7.2 in
/usr/local/lib/python3.10/dist-packages (from en-core-web-sm==3.7.1) (3.7.2)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in
/usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in
/usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (1.0.10)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in
/usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (2.0.8)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in
/usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (3.0.9)
Requirement already satisfied: thinc<8.3.0,>=8.1.8 in
/usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (8.2.3)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in
/usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
```

```
sm==3.7.1) (1.1.2)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in
/usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (2.4.8)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in
/usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (2.0.10)
Requirement already satisfied: weasel<0.4.0,>=0.1.0 in
/usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (0.3.4)
Requirement already satisfied: typer<0.10.0,>=0.3.0 in
/usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (0.9.0)
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in
/usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (6.4.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in
/usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (4.66.1)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in
/usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (2.31.0)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in
/usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (2.6.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
(from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (3.1.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-
packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (67.7.2)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (23.2)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in
/usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (3.3.0)
Requirement already satisfied: numpy>=1.19.0 in /usr/local/lib/python3.10/dist-
packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (1.25.2)
Requirement already satisfied: annotated-types>=0.4.0 in
/usr/local/lib/python3.10/dist-packages (from
pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (0.6.0)
Requirement already satisfied: pydantic-core==2.16.2 in
/usr/local/lib/python3.10/dist-packages (from
pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (2.16.2)
Requirement already satisfied: typing-extensions>=4.6.1 in
/usr/local/lib/python3.10/dist-packages (from
pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy<3.8.0,>=3.7.2->en-core-web-
```

```
sm==3.7.1) (4.9.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from
requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from
requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from
requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (2024.2.2)
Requirement already satisfied: blis<0.8.0,>=0.7.8 in
/usr/local/lib/python3.10/dist-packages (from
thinc<8.3.0,>=8.1.8->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (0.7.11)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in
/usr/local/lib/python3.10/dist-packages (from
thinc<8.3.0,>=8.1.8->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (0.1.4)
Requirement already satisfied: click<9.0.0,>=7.1.1 in
/usr/local/lib/python3.10/dist-packages (from
typer<0.10.0,>=0.3.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (8.1.7)
Requirement already satisfied: cloudpathlib<0.17.0,>=0.7.0 in
/usr/local/lib/python3.10/dist-packages (from
weasel<0.4.0,>=0.1.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (0.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->spacy<3.8.0,>=3.7.2->en-
core-web-sm==3.7.1) (2.1.5)
 Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
```

[3]:
```
!pip install pyspellchecker
```

```
Collecting pyspellchecker
  Downloading pyspellchecker-0.8.1-py3-none-any.whl (6.8 MB)
                           6.8/6.8 MB
19.1 MB/s eta 0:00:00
Installing collected packages: pyspellchecker
Successfully installed pyspellchecker-0.8.1
```

[4]:
```
import spacy
from spacy.matcher import Matcher
from spacy.tokens import Token
import pandas as pd
import numpy as np
from nltk.stem.porter import PorterStemmer
import os
```

```
import sys
from pathlib import Path
from typing import List
from sklearn.pipeline import Pipeline, FeatureUnion
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from collections import Counter
from xgboost import XGBClassifier
from sklearn.metrics import fbeta_score, make_scorer
from sklearn.model_selection import RandomizedSearchCV
from sklearn.base import BaseEstimator, TransformerMixin
from bs4 import BeautifulSoup
import re
from spellchecker import SpellChecker
import warnings
warnings.filterwarnings('ignore')
```

[5]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[6]:
```
base_folder = Path('/content/drive/MyDrive/NLP_HW/')
data_folder = base_folder/'HW_2'
```

[7]:
```
file =  data_folder/'spam.csv'
df = pd.read_csv(file, encoding = 'ISO-8859-1')
df = df[['v1','v2']]
df = df.rename(columns ={'v1': 'label', 'v2': 'message'})
```

[8]:
```
df.head()
```

[8]:

	label	message
0	ham	Go until jurong point, crazy.. Available only …
1	ham	Ok lar… Joking wif u oni…
2	spam	Free entry in 2 a wkly comp to win FA Cup fina…
3	ham	U dun say so early hor… U c already then say…
4	ham	Nah I don't think he goes to usf, he lives aro…

2 Taking a subset of the dataset and splitting it to create train and test datasets

[9]:
```
df_large = df.sample(frac = 0.4, random_state = 0)
df_large
```

[9]:
```
     label                                                message
4456   ham  Aight should I just plan to come up later toni…
690    ham                                  Was the farm open?
944    ham  I sent my scores to sophas and i had to do sec…
3768   ham  Was gr8 to see that message. So when r u leavi…
1189   ham  In that case I guess I'll see you at campus lodge
…    …                                                    …
2672   ham                        Super msg da:)nalla timing.
5076   ham  Guy, no flash me now. If you go call me, call …
3302   ham         It'll be tough, but I'll do what I have to
4225   ham  Ok thats cool. Its , just off either raglan rd…
5343   ham  No go. No openings for that room 'til after th…

[2229 rows x 2 columns]
```

[10]:
```
df_large['label'] = df_large['label'].map({'spam':1, 'ham':0}).astype(int)
```

[11]:
```
X = df_large['message'].values
y = df_large['label'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

[12]:
```
f2score = make_scorer(fbeta_score, beta=2)
```

[13]:
```
counter = Counter(y)
estimate = counter[0] / counter[1]
```

2.0.1 Feature Engineering

We will use the featurizer class that was provided in the lecture along with few additions to count exclamations and misspelled words. We will also include the definition for the Custom Preprocessor class since this is used in the featurizer class.

Spacy Preprocessor

[14]:
```
class SpacyPreprocessor(BaseEstimator, TransformerMixin):

    """
    A text preprocessor that utilizes spaCy for efficient and flexible NLP.␣
↪Designed as a part of a scikit-learn
    pipeline, it provides a wide range of text cleaning and preprocessing␣
↪functionalities.

    Attributes:
        model (str): The spaCy language model to be used for tokenization and␣
↪other NLP tasks.
        batch_size (int): The number of documents to process at once during␣
↪spaCy's pipeline processing.
```

```
        lemmatize (bool): If True, lemmatize tokens.
        lower (bool): If True, convert all characters to lowercase.
        remove_stop (bool): If True, remove stopwords.
        remove_punct (bool): If True, remove punctuation.
        remove_email (bool): If True, remove email addresses.
        remove_url (bool): If True, remove URLs.
        remove_num (bool): If True, remove numbers.
        stemming (bool): If True, apply stemming to tokens (mutually exclusive␣
↪with lemmatization).
        add_user_mention_prefix (bool): If True, add '@' as a separate token␣
↪(useful for user mentions in social
            media data).
        remove_hashtag_prefix (bool): If True, do not separate '#' from the␣
↪following text.
        basic_clean_only (bool): If True, perform only basic cleaning (HTML␣
↪tags removal, line breaks, etc.)
            and ignore other preprocessing steps.

    Methods:
        basic_clean(text: str) -> str:
            Performs basic cleaning of the text such as removing HTML tags and␣
↪excessive whitespace.

        spacy_preprocessor(texts: list) -> list:
            Processes a list of texts through the spaCy pipeline with specified␣
↪preprocessing options.

        fit(X, y=None) -> 'SpacyPreprocessor':
            Fits the preprocessor to the data. This is a dummy method for␣
↪scikit-learn compatibility and does not
            change the state of the object.

        transform(X, y=None) -> list:
            Transforms the provided data using the defined preprocessing␣
↪pipeline. Performs basic cleaning,
            and if `basic_clean_only` is False, it applies advanced spaCy␣
↪preprocessing steps.

    Raises:
        ValueError: If both 'lemmatize' and 'stemming' are set to True.
        ValueError: If 'basic_clean_only' is True but other processing options␣
↪are also set to True.
        TypeError: If the input X is not a list or a numpy array.
    """
```

```
  def __init__(self, model, *, batch_size = 64, lemmatize=True, lower=True,␣
↪remove_stop=True,
               remove_punct=True, remove_email=True, remove_url=True,␣
↪remove_num=False, stemming = False,
               add_user_mention_prefix=True, remove_hashtag_prefix=False,␣
↪basic_clean_only=False):

      self.model = model
      self.batch_size = batch_size
      self.remove_stop = remove_stop
      self.remove_punct = remove_punct
      self.remove_num = remove_num
      self.remove_url = remove_url
      self.remove_email = remove_email
      self.lower = lower
      self.add_user_mention_prefix = add_user_mention_prefix
      self.remove_hashtag_prefix = remove_hashtag_prefix
      self.basic_clean_only = basic_clean_only

      if lemmatize and stemming:
          raise ValueError("Only one of 'lemmatize' and 'stemming' can be␣
↪True.")

      # Validate basic_clean_only option
      if self.basic_clean_only and (lemmatize or lower or remove_stop or␣
↪remove_punct or remove_num or stemming or
                                    add_user_mention_prefix or␣
↪remove_hashtag_prefix):
          raise ValueError("If 'basic_clean_only' is set to True, other␣
↪processing options must be set to False.")

      # Assign lemmatize and stemming

      self.lemmatize = lemmatize
      self.stemming = stemming

  def basic_clean(self, text):
      soup = BeautifulSoup(text, "html.parser")
      text = soup.get_text()
      text = re.sub(r'[\n\r]', ' ', text)
      return text.strip()

  def get_cores(self):
      """
      Get the number of CPU cores to use in parallel processing.
      """
```

```
        # Get the number of CPU cores available on the system.
        num_cores = os.cpu_count()
        if num_cores < 3:
            use_cores = 1
        else:
            use_cores = num_cores // 2 + 1
        return use_cores

    def spacy_preprocessor(self, texts):
        final_result = []
        nlp = spacy.load(self.model)

        # Disable unnecessary pipelines in spaCy model
        if self.lemmatize:
            # Disable parser and named entity recognition
            disabled_pipes = ['parser', 'ner']
        else:
            # Disable tagger, parser, attribute ruler, lemmatizer and named␣
↪entity recognition
            disabled_pipes = ['tok2vec', 'tagger', 'parser', 'attribute_ruler',␣
↪'lemmatizer', 'ner']

        with nlp.select_pipes(disable=disabled_pipes):
          # Modify tokenizer behavior based on user_mention_prefix and␣
↪hashtag_prefix settings
          if self.add_user_mention_prefix or self.remove_hashtag_prefix:
              prefixes = list(nlp.Defaults.prefixes)
              if self.add_user_mention_prefix:
                  prefixes += ['@']  # Treat '@' as a separate token
              if self.remove_hashtag_prefix:
                  prefixes.remove(r'#')  # Don't separate '#' from the␣
↪following text
              prefix_regex = spacy.util.compile_prefix_regex(prefixes)
              nlp.tokenizer.prefix_search = prefix_regex.search

          # Process text data in parallel using spaCy's nlp.pipe()
          for doc in nlp.pipe(texts, batch_size=self.batch_size, n_process=self.
↪get_cores()):
              filtered_tokens = []
              for token in doc:
                  # Check if token should be removed based on specified filters
                  if self.remove_stop and token.is_stop:
                      continue
                  if self.remove_punct and token.is_punct:
                      continue
                  if self.remove_num and token.like_num:
                      continue
```

```
                if self.remove_url and token.like_url:
                    continue
                if self.remove_email and token.like_email:
                    continue

                # Append the token's text, lemma, or stemmed form to the␣
↪filtered_tokens list
                if self.lemmatize:
                    filtered_tokens.append(token.lemma_)
                elif self.stemming:
                    filtered_tokens.append(PorterStemmer().stem(token.text))
                else:
                    filtered_tokens.append(token.text)

            # Join the tokens and apply lowercasing if specified
            text = ' '.join(filtered_tokens)
            if self.lower:
                text = text.lower()
            final_result.append(text.strip())

        return final_result


    def fit(self, X, y=None):
        return self

    def transform(self, X, y=None):
        try:
            if not isinstance(X, (list, np.ndarray)):
                raise TypeError(f'Expected list or numpy array, got {type(X)}')

            x_clean = [self.basic_clean(text).encode('utf-8', 'ignore').
↪decode() for text in X]

            # Check if only basic cleaning is required
            if self.basic_clean_only:
                return x_clean  # Return the list of basic-cleaned texts

            x_clean_final = self.spacy_preprocessor(x_clean)
            return x_clean_final

        except Exception as error:
            print(f'An exception occurred: {repr(error)}')
```

####**Featurizer**

[15]:
```
class ManualFeatures(TransformerMixin, BaseEstimator):

    """A transformer class for extracting manual features from text data.

    This class is designed to be used in a scikit-learn pipeline. It uses the␣
↪spaCy
    library to extract a variety of manual features from text data, such as
    part-of-speech (POS) features, named entity recognition (NER) features,
    and count-based features.
    """



    def __init__(self, spacy_model='en_core_web_sm', batch_size = 64,␣
↪pos_features = True, ner_features = True, count_features = True):

        """
        Initialize the feature extractor.

        Parameters
        ----------
        spacy_model : str
            The name of the spaCy model to use for feature extraction.
        pos_features : bool, optional (default=True)
            Whether to extract part-of-speech (POS) features from the text data.
        ner_features : bool, optional (default=True)
            Whether to extract named entity recognition (NER) features from the␣
↪text data.
        count_features : bool, optional (default=True)
            Whether to extract count-based features from the text data.
        """

        self.spacy_model = spacy_model
        self.batch_size = batch_size
        self.pos_features = pos_features
        self.ner_features = ner_features
        self.count_features = count_features

    def get_cores(self):
        """
        Get the number of CPU cores to use in parallel processing.
        """
        # Get the number of CPU cores available on the system.
        num_cores = os.cpu_count()
        if num_cores < 3:
            use_cores = 1
        else:
```

```
            use_cores = num_cores // 2 + 1
        return num_cores

    def get_pos_features(self, cleaned_text):

        nlp = spacy.load(self.spacy_model)
        noun_count = []
        aux_count = []
        verb_count = []
        adj_count =[]

        # Disable the lemmatizer and NER pipelines for improved performance
        disabled_pipes = ['lemmatizer', 'ner']
        with nlp.select_pipes(disable=disabled_pipes):
            n_process = self.get_cores()
            for doc in nlp.pipe(cleaned_text, batch_size=self.batch_size,␣
↪n_process=n_process):
                # Extract nouns, auxiliaries, verbs, and adjectives from the␣
↪document
                nouns = [token.text for token in doc if token.pos_ in␣
↪["NOUN","PROPN"]]
                auxs =  [token.text for token in doc if token.pos_ in ["AUX"]]
                verbs =  [token.text for token in doc if token.pos_ in ["VERB"]]
                adjectives =  [token.text for token in doc if token.pos_ in␣
↪["ADJ"]]

                # Store the count of each type of word in separate lists
                noun_count.append(len(nouns))
                aux_count.append(len(auxs))
                verb_count.append(len(verbs))
                adj_count.append(len(adjectives))

        # Stack the count lists vertically to form a 2D numpy array
        return np.transpose(np.vstack((noun_count, aux_count, verb_count,␣
↪adj_count)))



    def get_ner_features(self, cleaned_text):
        nlp = spacy.load(self.spacy_model)
        count_ner = []

        # Disable the tok2vec, tagger, parser, attribute ruler, and lemmatizer␣
↪pipelines for improved performance
        disabled_pipes = ['tok2vec', 'tagger', 'parser', 'attribute_ruler',␣
↪'lemmatizer']
```

```
        with nlp.select_pipes(disable=disabled_pipes):
            n_process = self.get_cores()
            for doc in nlp.pipe(cleaned_text, batch_size=self.batch_size,␣
↪n_process=n_process):
                ners = [ent.label_ for ent in doc.ents]
                count_ner.append(len(ners))

        # Convert the list of NER counts to a 2D numpy array
        return np.array(count_ner).reshape(-1, 1)


    def get_count_features(self, cleaned_text):
        list_count_words = []
        list_count_characters = []
        list_count_characters_no_space = []
        list_avg_word_length = []
        list_count_digits = []
        list_count_numbers = []
        list_count_misspell=[]
        list_count_sentences = []

        nlp = spacy.load(self.spacy_model)
        disabled_pipes = ['tok2vec', 'tagger', 'parser', 'attribute_ruler',␣
↪'lemmatizer', 'ner']
        with nlp.select_pipes(disable=disabled_pipes):
            if not nlp.has_pipe('sentencizer'):
                nlp.add_pipe('sentencizer')
            n_process = self.get_cores()
            for doc in nlp.pipe(cleaned_text, batch_size=self.batch_size,␣
↪n_process=n_process):
                count_word = len([token for token in doc if not token.is_punct])
                count_char = len(doc.text)
                count_char_no_space = len(doc.text_with_ws.replace(' ', ''))
                avg_word_length = count_char_no_space / (count_word + 1)
                count_numbers = len([token for token in doc if token.is_digit])
                count_sentences = len(list(doc.sents))

                list_count_words.append(count_word)
                list_count_characters.append(count_char)
                list_count_characters_no_space.append(count_char_no_space)
                list_avg_word_length.append(avg_word_length)
                list_count_numbers.append(count_numbers)
                list_count_sentences.append(count_sentences)

        count_features = np.vstack((list_count_words, list_count_characters,␣
↪list_count_characters_no_space, list_avg_word_length,
                                    list_count_numbers, list_count_sentences))
```

```
        return np.transpose(count_features)


    def fit(self, X, y=None):
        """
        Fit the feature extractor to the input data.

        This method does not actually do any fitting, as the feature extractor␣
↪is stateless.
        It simply returns the instance of the class.

        Parameters:
        X (list or numpy.ndarray): The input data.
        y (list or numpy.ndarray, optional): The target labels. Not used in␣
↪this implementation.

        Returns:
        FeatureExtractor: The instance of the class.
        """
        return self


    def transform(self, X, y=None):
        """
        Transform the input data into a set of features.

        Parameters:
        X (list or numpy.ndarray): The input data.
        y (list or numpy.ndarray, optional): The target labels. Not used in␣
↪this implementation.

        Returns:
        tuple: A tuple containing a 2D numpy array with shape (len(X),␣
↪num_features) where num_features is the number of features extracted and a␣
↪list of feature names.

        Raises:
        TypeError: If the input data is not a list or numpy array.
        Exception: If an error occurs while transforming the data into features.
        """
        try:
            # Check if the input data is a list or numpy array
            if not isinstance(X, (list, np.ndarray)):
                raise TypeError(f"Expected list or numpy array, got {type(X)}")

            preprocessor1 = SpacyPreprocessor(model='en_core_web_sm',␣
↪batch_size=64, lemmatize=False, lower=False,
```

```
                                    remove_stop=False, remove_email=True,
                                    remove_url=True, remove_num=False,␣
↪stemming=False,
                                    add_user_mention_prefix=True,␣
↪remove_hashtag_prefix=False, basic_clean_only=False)
            preprocessor2 = SpacyPreprocessor(model='en_core_web_sm',␣
↪batch_size=64, lemmatize=False, lower=False,
                                    remove_stop=False, remove_punct=False,␣
↪remove_email=True,
                                    remove_url=True, remove_num=False,␣
↪stemming=False,
                                    add_user_mention_prefix=True,␣
↪remove_hashtag_prefix=False, basic_clean_only=False)

            feature_names = []
            if self.pos_features or self.ner_features:
                cleaned_x_count_ner_pos = preprocessor2.fit_transform(X)

            if self.count_features:
                cleaned_x_count_features = preprocessor1.fit_transform(X)
                count_features = self.
↪get_count_features(cleaned_x_count_features)
                feature_names.extend(['count_words', 'count_characters',
                                      'count_characters_no_space',␣
↪'avg_word_length',
                                      'count_numbers', 'count_sentences'])
            else:
                count_features = np.empty(shape=(0, 0))

            if self.pos_features:
                pos_features = self.get_pos_features(cleaned_x_count_ner_pos)
                feature_names.extend(['noun_count', 'aux_count', 'verb_count',␣
↪'adj_count'])
            else:
                pos_features = np.empty(shape=(0, 0))

            if self.ner_features:
                ner_features = self.get_ner_features(cleaned_x_count_ner_pos)
                feature_names.extend(['ner'])
            else:
                ner_features = np.empty(shape=(0, 0))

           # Stack the feature arrays horizontally to form a single 2D numpy␣
↪array
            if ner_features.shape == (0, 0) and pos_features.shape == (0, 0):
              return np.hstack((count_features))
```

```
            elif pos_features.shape == (0, 0):
              return np.hstack((count_features, ner_features))
            elif ner_features.shape == (0, 0):
              return np.hstack((count_features, pos_features))
            else:
              return np.hstack((count_features, ner_features, pos_features))

        except Exception as error:
            print(f'An exception occured: {repr(error)}')
```

3 Chosen Pipeline -> Sparse Embeddings + Feature Engineering

We are using the hyperparameters as per our results from the cross validation

[16]:
```
feature_extract = FeatureUnion([("sparse_embed",␣
 ↪TfidfVectorizer(max_features=100, max_df=0.75)), ('feature_eng',␣
 ↪ManualFeatures(pos_features=True, ner_features = False))])
```

[17]:
```
final_pipe = Pipeline([('fe', feature_extract), ('classifier',␣
 ↪XGBClassifier(scale_pos_weight=estimate))])
```

[18]:
```
final_pipe.fit(X_train, y_train)
```

[18]:
```
Pipeline(steps=[('fe',
                 FeatureUnion(transformer_list=[('sparse_embed',
                                                 TfidfVectorizer(max_df=0.75,
max_features=100)),
                                                ('feature_eng',
ManualFeatures(ner_features=False))])),
                ('classifier',
                 XGBClassifier(base_score=None, booster=None, callbacks=None,
                               colsample_bylevel=None, colsample_bynode=None,
                               colsample_bytree=None, device=None,
                               early_stopping_rounds=None,
                               en…
                               feature_types=None, gamma=None, grow_policy=None,
                               importance_type=None,
                               interaction_constraints=None, learning_rate=None,
                               max_bin=None, max_cat_threshold=None,
                               max_cat_to_onehot=None, max_delta_step=None,
                               max_depth=None, max_leaves=None,
                               min_child_weight=None, missing=nan,
                               monotone_constraints=None, multi_strategy=None,
                               n_estimators=None, n_jobs=None,
                               num_parallel_tree=None, random_state=None,
…))])
```

[19]:
```
y_test_pred = final_pipe.predict(X_test)
```

[20]:
```
print("The test score of the final pipeline is", fbeta_score(y_test,␣
↪y_test_pred, beta=2))
```

```
The test score of the final pipeline is 0.8815426997245178
```

Running the final pipeline on the large dataset (40% of the dataset), we get a test score of 0.8815426997245178

[ ]:
```

```