practical-3-mids

April 1, 2025

[1]:

**import nltk**

**from nltk.corpus import** stopwords

**from nltk.tokenize import** word\_tokenize

**from nltk.classify import** NaiveBayesClassifier

**from nltk.classify.util import** accuracy

**import string**

[2]:

*# Download required NLTK resources* nltk.download('punkt') nltk.download('stopwords')

[nltk\_data] Downloading package punkt to

[nltk\_data] C:\Users\NIKKI\AppData\Roaming\nltk\_data… [nltk\_data] Package punkt is already up-to-date! [nltk\_data] Downloading package stopwords to

[nltk\_data] C:\Users\NIKKI\AppData\Roaming\nltk\_data… [nltk\_data] Package stopwords is already up-to-date!

[2]: True

[8]:

*# Define positive and negative word sets globally*

positive\_words = set(["good", "happy", "excellent", "great", "love", "awesome",␣

𝗌"fantastic", "positive", "best", "joy"])

negative\_words = set(["bad", "sad", "terrible", "awful", "hate", "worst",␣

𝗌"negative", "horrible", "disgusting", "angry"])

[9]:

*# Preprocessing function*

**def** preprocess\_text(text):

text = text.lower() *# Convert to lowercase*

text = text.translate(str.maketrans('', '', string.punctuation)) *# Remove*␣

𝗌*punctuation*

words = word\_tokenize(text) *# Tokenization*

stop\_words = set(stopwords.words("english")) *# Load stopwords*

words = [word **for** word **in** words **if** word **not in** stop\_words] *# Remove*␣

𝗌*stopwords*

**return** words

[10]:

*# Feature extraction function*

**def** extract\_features(words): features = {}

**for** word **in** positive\_words: features[f"contains(**{**word**}**)"] = (word **in** words)

**for** word **in** negative\_words: features[f"contains(**{**word**}**)"] = (word **in** words)

**return** features

[11]:

*# Training data*

train\_data = [

("I love this product, it is awesome!", "Positive"),

("This is the worst thing I have ever bought!", "Negative"), ("I am very happy with this service.", "Positive"),

("I hate this, it is terrible!", "Negative"),

("This is fantastic, I highly recommend it!", "Positive"), ("I am so sad about this experience.", "Negative"),

]

[12]:

train\_set = [(extract\_features(preprocess\_text(text)), label) **for** text, label␣

𝗌**in** train\_data]

[13]:

*# Train Naive Bayes Classifier*

classifier = NaiveBayesClassifier.train(train\_set)

[14]:

*# Classification function*

**def** classify\_text(text):

words = preprocess\_text(text) features = extract\_features(words) **return** classifier.classify(features)

[15]:

*# Test classification*

new\_text = "I am feeling very happy and excited today!" print(f"Sentiment: **{**classify\_text(new\_text)**}**")

Sentiment: Positive

[16]:

new\_text2 = "This is a terrible experience. I hate it!" print(f"Sentiment: **{**classify\_text(new\_text2)**}**")

Sentiment: Negative

[ ]: