NAAN MUDHALVAN

ARTIFICIAL INTELLIGENCE

PROJECT TITLE

SENTIMENT ANALYSIS FOR MARKETING

REG NO: 712321104010
NAME: PREMANANTH.P
DEPT: COMPUTERSCIENCE&ENGINEERING
YEAR & SEM: 3rd & 5th
COLLEGE NAME: PARK COLLEGE OF ENGINEERING AND TECHNOLOGY

PHASE 3

INTRODUCTION:

- Sentiment analysis studies people's sentiments in their produced text, such as product reviews, blog comments, and forum discussions. It enjoys wide applications to fields as diverse as politics (e.g., analysis of public sentiments towards policies), finance (e.g., analysis of sentiments of the market), and marketing (e.g., product research and brand management).

- Since sentiments can be categorized as discrete polarities or scales (e.g., positive and negative), we can consider sentiment analysis as a text classification task, which transforms a varying-length text sequence into a fixed-length text category. In this chapter, we will use Stanford's large movie review dataset for sentiment analysis. It consists of a training set and a testing set, either containing 25000 movie reviews downloaded from IMDb. In both datasets, there are equal number of "positive" and "negative" labels, indicating different sentiment polarities.

Loading the Dataset :

```
#@saved2l.DATA_HUB['aclImdb'] = (d2l.DATA_URL +
'aclImdb_v1.tar.gz',

'01ada507287d82875905620988597833ad4e0903')
data_dir = d2l.download_extract('aclImdb', 'aclImdb')
```

Next, read the training and test datasets. Each example is a review and its label: 1 for “positive” and 0 for “negative” .

```
#@savedef read_imdb(data_dir, is_train):    """Read the IMDb
review dataset text sequences and labels."""
    data, labels = [], []
    for label in ('pos', 'neg'):
        folder_name = os.path.join(data_dir, 'train' if is_train else
'test',
                                   label)
        for file in os.listdir(folder_name):
            with open(os.path.join(folder_name, file), 'rb') as f:
                review = f.read().decode('utf-
8').replace('\n', '')
                 data.append(review)
                labels.append(1 if label ==
'pos' else 0)
    return data, labels
train_data = read_imdb(data_dir,
is_train=True)print('# trainings:',
len(train_data[0]))for x, y in
zip(train_data[0][:3], train_data[1][:3]):
    print('label:', y, 'review:', x[:60])
```

Program:

```
from vaderSentiment.vaderSentiment import
SentimentIntensityAnalyzer

sentiment = SentimentIntensityAnalyzer()

text_1 = "The book was a perfect balance between wrtiting
style and plot."

text_2 =  "The pizza tastes terrible."

sent_1 = sentiment.polarity_scores(text_1)

sent_2 = sentiment.polarity_scores(text_2)

print("Sentiment of text 1:", sent_1)

print("Sentiment of text 2:", sent_2)
```

Output:

```
Sentiment of text 1: {'neg': 0.0, 'neu': 0.73, 'pos': 0.27,
'compound': 0.5719}

Sentiment of text 2: {'neg':0.508, 'neu ':0.492, 'pos ':0.0,
'compound ': -0.4767}
```

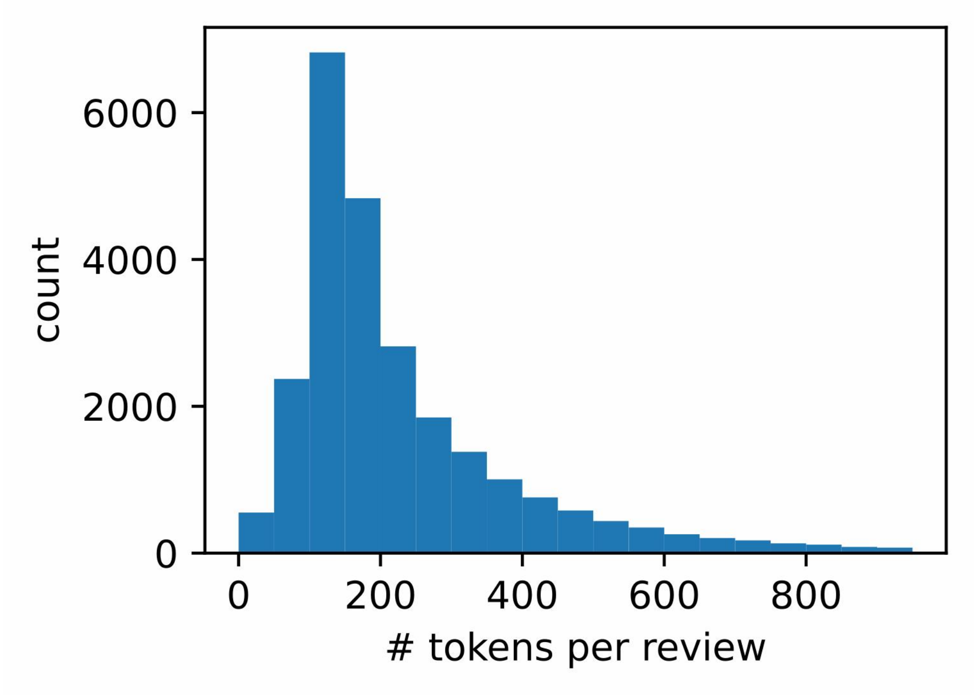
Preprocessing the dataset:

Treating each word as a token and filtering out words that appear less than 5 times, we create a vocabulary out of the training dataset.

```
train_tokens=d2l.tokenize(train_data[0],token='word')vocab=d2l.Vocab(train_tokens, min_freq=5, reserved_tokens=['<pad>'])
```

Plot the histogram of review lengths in tokens.

```
d2l.set_figsize()d2l.plt.xlabel('# tokens per review')d2l.plt.ylabel('count')d2l.plt.hist([len(line) for line in train_tokens], bins=range(0, 1000, 50));
```



```
num_steps = 500  # sequence lengthtrain_features =
torch.tensor([d2l.truncate_pad(
   vocab[line], num_steps, vocab['<pad>']) for line in
train_tokens])print(train_features.shape)
```

```
torch.Size([25000, 500])
```

CHALLENGES INVOLVED IN LOAD &PREPROCESSING THE DATASET IN SENTIMENT ANALYSIS:

1.Dealing with ambiguous language:

One of the most significant challenges sentiment analysis models face is handling ambiguous language. Ambiguity arises from various factors, including sarcasm, idioms, double meanings, and irony. For instance, the phrase "I can't wait" could express positive anticipation or sarcastic disappointment, making it difficult for a model to determine the correct sentiment.

2.Handling negation:

Another challenge that sentiment analysis models face is managing negation. Negation can significantly alter the meaning of a sentence, turning a positive sentiment into a negative one or vice versa. For example, the phrase "not bad" actually means "good," and it can be difficult for a model to understand this reversal of sentiment.

3.Adapting to different domains and industries:

Sentiment analysis models often struggle to adapt to different domains and industries. Each industry has its own unique jargon, slang, and terminology, which can significantly influence sentiment interpretation. For example, the term "sick" has a negative connotation in the healthcare industry but can be used positively to describe something cool or impressive in informal settings.

4.Coping with multilingual data:

As businesses expand globally, they encounter multilingual data from customers across different regions. Most sentiment analysis models are trained on a single language, making it challenging to analyze text data in multiple languages accurately. But doing so leaves out valuable qualitative data in other languages that can go a long way in helping businesses understand how customers in different regions feel about their offerings.

Overcome these challenges with sentiment analysis tools powered by AI:

Sentiment analysis models offer businesses a powerful way to understand and respond to their customers' emotions and opinions. However, traditional models face various challenges that prevent companies from gaining a holistic view of their customers' feelings towards their offerings.

By leveraging generative analysis tools from Viable, businesses can overcome these challenges and harness the full potential of sentiment analysis to make more informed decisions and improve customer satisfaction.