Amazon Fine Food Reviews - Sentiment Analysis

312706034 資管碩二 張棨翔

1. TF-IDF

TF-IDF（Term Frequency-Inverse Document Frequency）在情感分析（sentiment analysis）中的作用主要是用來評估文本中單詞的重要性。這種方法可以幫助分析和理解文本資料，尤其是在處理大量的評論或社交媒體訊息時。

- 詞頻（TF, Term Frequency）
  - 衡量某個詞在一篇文本中的出現頻率。
  - 表示某個詞在該文本中的重要性，出現越頻繁的詞其TF值越高。
- 逆文檔頻率（IDF, Inverse Document Frequency）
  - 衡量某個詞在整個文檔集中的重要性。
  - 如果一個詞在很多文檔中出現，它的IDF值會低，反之則高。
- TF-IDF計算
  - 可以得到一個詞對於特定文檔的相對重要性。

1. 載入需要用到的套件
2. 讀入檔案並且只保留前10,000筆，然後去除不需要用到的欄位
3. 將Score欄位內的數值根據需求做對應的轉換
4. 將Text欄位內的句子進行分割
5. 移除對於情感分析不重要的文本中的Stop Words

```
import pandas as pd
import nltk
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score

df = pd.read_csv("Reviews.csv")

#only preserve the first 10000 rows
df = df[:10000]

#only preserve the 'Score' and 'Text' columns
df = df[['Score', 'Text']]

# Convert the values in the "Score" column that are greater than or equal to 4 to 1, and the rest to 0 (1: positive, 0: neg
df['Score'] = df['Score'].apply(lambda x: 1 if x >= 4 else 0)

#Split the text in the "Text" column using a delimiter
df['Text'] = df['Text'].str.split()

#Remove stop words
nltk.download('stopwords')
stop_words = stopwords.words('english')
df['Text'] = df['Text'].apply(lambda x: [word for word in x if word not in stop_words])
```

6. 透過TF-IDF將Text欄位當中的資料轉換成向量
7. 建立隨機森林模型用來做情感分析的訓練

8. 透過K-fold validation來驗證模型在accuracy上的表現(k=4)

```
#Text mining preprocessing, converting text into vectors, implement tf-idf (sklearn.feature_extraction.text.TfidfVectorizer)
tfidf = TfidfVectorizer()
df['Text'] = df['Text'].apply(lambda x: ' '.join(x))

#Apply tf-idf to the "Text" column
tfidf_matrix = tfidf.fit_transform(df['Text'])
tfidf_matrix

#Use Random Forest Classifier (TF-IDF)
clf = RandomForestClassifier()
clf.fit(tfidf_matrix, df['Score'])

#Perform k-fold cross-validation and calculate the accuracy for k=4
scores = cross_val_score(clf, tfidf_matrix, df['Score'], cv=4, scoring='accuracy')
print(f'Cross-validation scores: {scores}')
print(f'Average accuracy: {scores.mean():.4f}')
```

9. 印出4次Validation的accuracy並得到平均accuracy = 0.7943

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\win7-006\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
Cross-validation scores: [0.7904 0.792  0.7936 0.8012]
Average accuracy: 0.7943
```

---

2. Word2Vec

Word2Vec 是一種流行的詞嵌入技術，用於將單詞轉換為向量表示。在情感分析中，Word2Vec 的作用主要是捕捉單詞之間的語義關係，幫助模型更好地理解文本內容。而 Word2Vec 通常有兩種模型：CBOW（Continuous Bag of Words）和 Skip-gram。

- CBOW
  - 根據上下文詞語預測中心詞，試圖透過周圍的詞來預測某個特定詞。
- Skip-Gram
  - 從中心詞預測周圍的詞，這種方法更適合處理小型語料庫，並能捕捉到稀有詞的語義。

在訓練過程中，Word2Vec 會考慮一定範圍的上下文詞，並利用這些上下文來訓練詞向量，使得語義相近的詞在向量空間中距離較近。訓練後，每個詞都會被轉換成一個固定維度的向量，這些向量能夠捕捉到詞與詞之間的語義關係。

1. 載入需要用到的套件
2. 讀入檔案並且只保留前10,000筆，然後去除不需要用到的欄位
3. 將Score欄位內的數值根據需求做對應的轉換
4. 將Text欄位內的句子進行分割

5. 移除對於情感分析不重要的文本中的Stop Words

```
import pandas as pd
import nltk
from nltk.corpus import stopwords
from gensim.models import Word2Vec
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
import numpy as np

# Load the dataset
df = pd.read_csv("Reviews.csv")

# Preserve only the first 10000 rows and select 'Score' and 'Text' columns
df = df[['Score', 'Text']].head(10000)

# Convert 'Score' to binary (1 for positive, 0 for negative sentiment)
df['Score'] = df['Score'].apply(lambda x: 1 if x >= 4 else 0)

# Download necessary NLTK resources
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))

# Preprocess the text (tokenization and stopword removal)
df['Text'] = df['Text'].apply(lambda x: [word for word in x.lower().split() if word not in stop_words])
```

6. 透過Word2Vec將Text欄位當中的資料轉換成向量
7. 建立隨機森林模型用來做情感分析的訓練
8. 透過K-fold validation來驗證模型在accuracy上的表現(k=4)

```
# Train Word2Vec model on the tokenized text
w2v = Word2Vec(df['Text'], vector_size=100, window=5, min_count=2)

# Compute the average Word2Vec vector for each document
def document_vector(text):
    # Filter words that are in the Word2Vec model's vocabulary
    words = [word for word in text if word in w2v.wv]
    if len(words) == 0:  # If none of the words are in the vocabulary, return a zero vector
        return np.zeros(100)
    # Average the word vectors
    return np.mean(w2v.wv[words], axis=0)

# Apply the function to create a document vector for each review
w2v_matrix = np.vstack(df['Text'].apply(document_vector))

# Use Random Forest Classifier on the Word2Vec vectors
clf = RandomForestClassifier()
clf.fit(w2v_matrix, df['Score'])

# Perform 4-fold cross-validation and print the accuracy
scores = cross_val_score(clf, w2v_matrix, df['Score'], cv=4, scoring='accuracy')
print(f'Cross-validation scores: {scores}')
print(f'Average accuracy: {scores.mean():.4f}')
```

9. 印出4次Validation的accuracy並得到平均accuracy = 0.7611

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\win7-006\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
Cross-validation scores: [0.7676 0.7624 0.7548 0.7596]
Average accuracy: 0.7611
```

TF-IDF與Word2Vec比較

- TF-IDF（Term Frequency-Inverse Document Frequency）

    - 優點：簡單直觀，易於計算和實現。在文本分類中效果良好，尤其是當重要的關鍵詞對情感判斷至關重要時。
    - 缺點：無法捕捉詞與詞之間的語義關係，相似的詞會被視為完全不同的特徵，對於上下文敏感性較低。
- Word2Vec
    - 優點：能夠捕捉語義關係和上下文信息，提供更豐富的特徵表示。對於處理同義詞、近義詞效果更好，能更好地反映文本中的情感。
    - 缺點：運算成本較高，尤其是在大規模語料庫上訓練模型時。對於短文本或特殊領域的文本可能需要進行額外調整或訓練。
- 兩者在應用上的比較
    - 準確性：Word2Vec 通常在情感分析中表現更佳，因為它能捕捉到詞之間的細微語義差異，這對於情感判斷至關重要。
    - 解釋性：TF-IDF 更容易解釋，因為它提供了每個詞的重要性指標。這對於需要理解為什麼某些詞會影響情感判斷的場合非常有用。
    - 特徵維度：TF-IDF 會生成一個高維稀疏矩陣，而 Word2Vec 生成的是低維密集向量，這對於後續的機器學習模型訓練更為高效。

在情感分析中，選擇 TF-IDF 還是 Word2Vec 取決於具體的應用場景和需求。如果需要簡單、快速且易於解釋的模型，TF-IDF 是不錯的選擇；而如果需要更深層次的語義理解，特別是在大型語料庫中，Word2Vec 將是更好的選擇。