Twitter Sentiment Analysis

A Student Project by
Mahmoud Abdelhamid Attia
Student ID: 20194239

Supervised by
Prof. Osama Fathy
TA: Mahinda Mahmoud Samy

Department of Computer Science

Future University

1.Abstract:
This documentation presents a sentiment analysis project using the K-Nearest Neighbors (KNN) algorithm for Twitter data. It covers problem definition, objectives, methodology, implementation, test results, and discussion. The chosen KNN methodology is justified, and implementation details are provided. The test results show an accuracy of 41%. The discussion reflects on the project's experience, strengths, weaknesses, challenges, and potential improvements. Overall, this documentation serves as a concise reference for sentiment analysis projects.

2. Problem Definition and Context/Background: Twitter sentiment analysis is a valuable technique that allows us to analyze the sentiment or opinion expressed in tweets. With the increasing popularity of social media platforms like Twitter, sentiment analysis provides insights into public opinions, customer feedback, and brand reputation. Sentiment analysis has applications in various domains, including market research, brand management, and social trend analysis. This project aims to develop a system that can accurately classify tweets into positive, negative, or neutral sentiments.

3. Project Objectives: The specific tasks that the system will perform include:
Collecting a dataset of tweets with associated sentiment labels
Preprocessing the tweet data by cleaning, tokenizing, removing stop words, and lemmatizing
Implementing a machine learning model, specifically the K-Nearest Neighbors (KNN) algorithm, for sentiment classification
Training the KNN model using the preprocessed tweet data
Evaluating the model's performance by testing it on a separate dataset
Interacting with the user by providing a real-time sentiment analysis for specific keywords or hashtags

4. Intelligent System Methodology and Justification:
The chosen methodology for this project is the K-Nearest Neighbors (KNN) algorithm. KNN is a non-parametric classification algorithm widely used for text classification tasks. It is suitable for sentiment analysis as it does not make strong assumptions about the underlying data distribution and can handle high-dimensional data effectively. KNN works by classifying a new data point based on the labels of its neighboring data points in the feature space.

5. Description of Implementation:

Software Tools: The implementation will utilize Python programming language and relevant libraries such as Pandas, scikit-learn, and NLTK.

Code:

```
import pandas as pd
import regex as re
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import nltk
```

Knowledge Acquisition: The dataset of tweets with sentiment labels will be acquired from a CSV file.

Code:

```
# TSA dataset (size= 162980)
dataset = pd.read_csv('Twitter_Data.csv')
```

the dataset from: https://www.kaggle.com/datasets/saurabhshahane/twitter-sentiment-dataset

Data Preprocessing: The tweet text will be preprocessed by converting it to lowercase, removing special characters, tokenizing the text, removing stop words, and lemmatizing the tokens.

Code:

```
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()
def preprocess(text):
    # convert to lowercase
    text = text.lower()
    # clean text
    text = re.sub(r'[^ a-z]',",text)
    # word Tokenizer
    tokens = word_tokenize(text)
    # remove stop words
    tokens = [token for token in tokens if token not in stop_words]
    # lemmatize tokens
    lemmTokens = [lemmatizer.lemmatize(token) for token in tokens]
```

```
    # join tokens
    text = " ".join(lemmTokens)
    return text
```

Training and Testing: The dataset will be split into training and testing sets using the train_test_split function from scikit-learn.

Code:

```
X_train, X_test, y_train, y_test = train_test_split(dataset.iloc[:,:1],
dataset.iloc[:,1:3], test_size=0.2)
```

The TF-IDF vectorizer will be used to convert the preprocessed text data into numerical features.

Code:

```
# Create the TF-IDF vectorizer
vectorizer = TfidfVectorizer()
# Fit the vectorizer with the training data
vectorizer.fit(X_train["clean_text"])

# Transform the training data
X_train_vectors = vectorizer.transform(X_train["clean_text"])

# Transform the testing data
X_test_vectors = vectorizer.transform(X_test["clean_text"])
```

The KNN classifier will be trained on the training data and
Reasoning Strategy: The KNN algorithm will be used for sentiment classification, where the sentiment labels of the nearest neighbors will be used to predict the sentiment of a new tweet.

Code:

```
# Create the k-NN classifier
knn = KNeighborsClassifier(n_neighbors=3)
# Train the classifier
knn.fit(X_train_vectors, y_train)
```

Testing:
evaluation on the testing data using accuracy as the evaluation metric.

Code:

```
from sklearn.metrics import accuracy_score
#predicting
y_pred = knn.predict(X_test_vectors)
```

6. Test Results: The accuracy of the sentiment analysis model on the test dataset is 41% .

Code:

```
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
```

7. Conclusion:

In conclusion, this documentation highlights a sentiment analysis project using the K-Nearest Neighbors (KNN) algorithm for Twitter data. The project aimed to analyze the sentiment of tweets and achieved an accuracy of 41%. The chosen methodology, KNN, demonstrated its effectiveness in classifying sentiments. However, the project also encountered challenges and identified areas for improvement. Overall, this documentation serves as a valuable resource for understanding sentiment analysis and provides insights into the implementation and evaluation of the project.