IQGateway Machine Learning Internship Task

Candidate Name: Keerthana Ramesh Babu

Email ID: priyamna04@gmail.com

HKBK College of Engineering, VTU

Task Chosen: EmoINT

(Here I have chosen and used the training, test and development dataset on the emotion- anger)

Existing emotion datasets are mainly annotated categorically without an indication of degree of emotion. Further, the tasks are almost always framed as classification tasks (identify 1 among n emotions for this sentence). In contrast, it is often useful for applications to know the degree to which an emotion is expressed in text. This is the first task where systems have to automatically determine the intensity of emotions in tweets.

Task:

Given a tweet and an emotion X, determine the intensity or degree of emotion X felt by the speaker -- a real-valued score between 0 and 1. The maximum possible score 1 stands for feeling the maximum amount of emotion X (or having a mental state maximally inclined towards feeling emotion X). The minimum possible score 0 stands for feeling the least amount of emotion X (or having a mental state maximally away from feeling emotion X). The tweet along with the emotion X will be referred to as an instance.

ABSTRACT:

Emotion intensity identification is a crucial task in natural language processing and affective computing, aimed at understanding the degree or intensity of emotions expressed in text. In this problem, given a piece of text and an emotion label, the objective is to predict the intensity of the specified emotion felt by the speaker on a continuous scale ranging from 0 to 1. This task is particularly challenging due to the subjective and nuanced nature of emotions, as well as the variability in expression styles across individuals. Effective solutions to this problem have wide-ranging applications in sentiment analysis, opinion mining, human-computer interaction, and personalized content recommendation systems.

OBJECTIVES:

Based on my research and understanding of the task, I have worked on to list out and accomplish the following objectives:

- The primary objective is to design and implement machine learning algorithms capable of accurately predicting the intensity of emotions expressed in text.
- To enhance the performance of the models, various feature engineering techniques such as TF-IDF (Term Frequency-Inverse Document Frequency), word embedding, and linguistic features.
- To evaluate the model performance using appropriate evaluation metrics such as Root Mean Squared Error (RMSE), correlation coefficient, and Spearman rank correlation.
- To identify the models that demonstrate the highest accuracy and generalization capability across different datasets.
- In addition to traditional machine learning models, exploring deep learning architectures such as recurrent neural networks (RNNs), long short-term memory networks (LSTMs) etc.
- To address domain-specific challenges such as data sparsity, cultural variations in expression, and subjective interpretation of intensity levels.

DESCRIPTION ABOUT THE DATASET:

I have used the following URLs to obtain training, test and development dataset for the emotion- anger, as assigned:

Training Data: https://saifmohammad.com/WebDocs/EmoInt%20Dev%20Data/anger-ratings-0to1.dev.target.txt

Test Data: https://saifmohammad.com/WebDocs/EmoInt%20Test%20Data/anger-ratings-0to1.test.target.txt

Development Data: https://saifmohammad.com/WebDocs/EmoInt%20Dev%20Data/anger-ratings-0to1.dev.target.txt

METHODOLOGY USED:

My approach to solve this particular task is to first apply various machine learning algorithms that I am familiar with, the calculate their scores, mean squared errors, as an additional metrics I have also included code for calculating the correlation score and Spearman Rank Coefficient too.

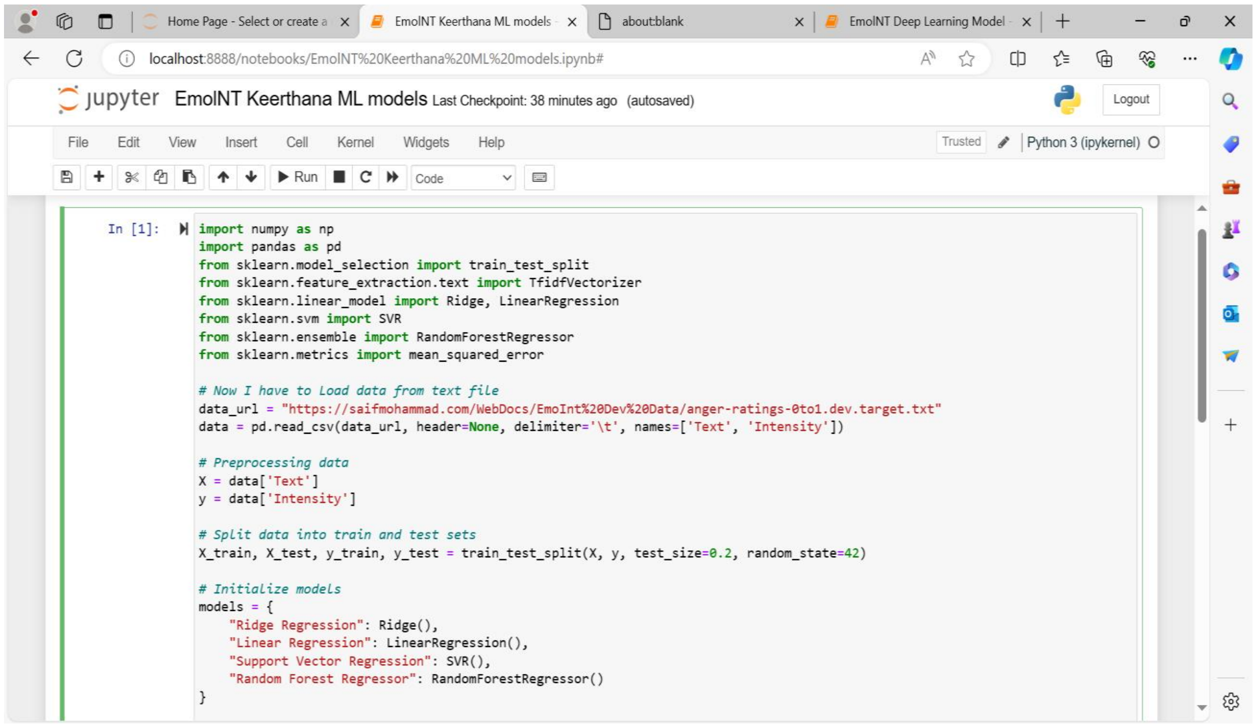
Additionally, I have also developed a deep learning model using an Embedding layer followed by an LSTM layer and a dense layer which tokenizes the text data and then trains and evaluates the model.

The steps I followed to accomplish the task are as follows:

1. **Data Collection**: Gather a dataset containing text samples annotated with the intensity of different emotions. Emotion intensity datasets are available for various emotions such as anger, joy, sadness, etc.
2. **Data Preprocessing**: Preprocess the text data by removing noise, such as special characters, punctuation, and stop words. Additionally, perform tokenization, stemming or lemmatization, and lowercasing to standardize the text format.
3. **Feature Extraction**: Extract relevant features from the preprocessed text data. Common feature extraction techniques include TF-IDF (Term Frequency-Inverse Document Frequency), word embeddings, and linguistic features (e.g., sentiment scores, part-of-speech tags).
4. **Model Selection**: Choose suitable machine learning algorithms for emotion intensity identification. Common algorithms include linear regression, ridge regression, support vector regression, random forest regression, and neural networks.
5. **Model Training**: Train the selected machine learning models on the feature vectors extracted from the training data.
6. **Model Evaluation**: Evaluate the trained models using appropriate evaluation metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Pearson correlation coefficient, and Spearman rank correlation coefficient.
7. **Model Selection**: Compare the performance of different machine learning models based on evaluation metrics. Select the model that achieves the highest accuracy or correlation coefficient on the validation or test dataset.

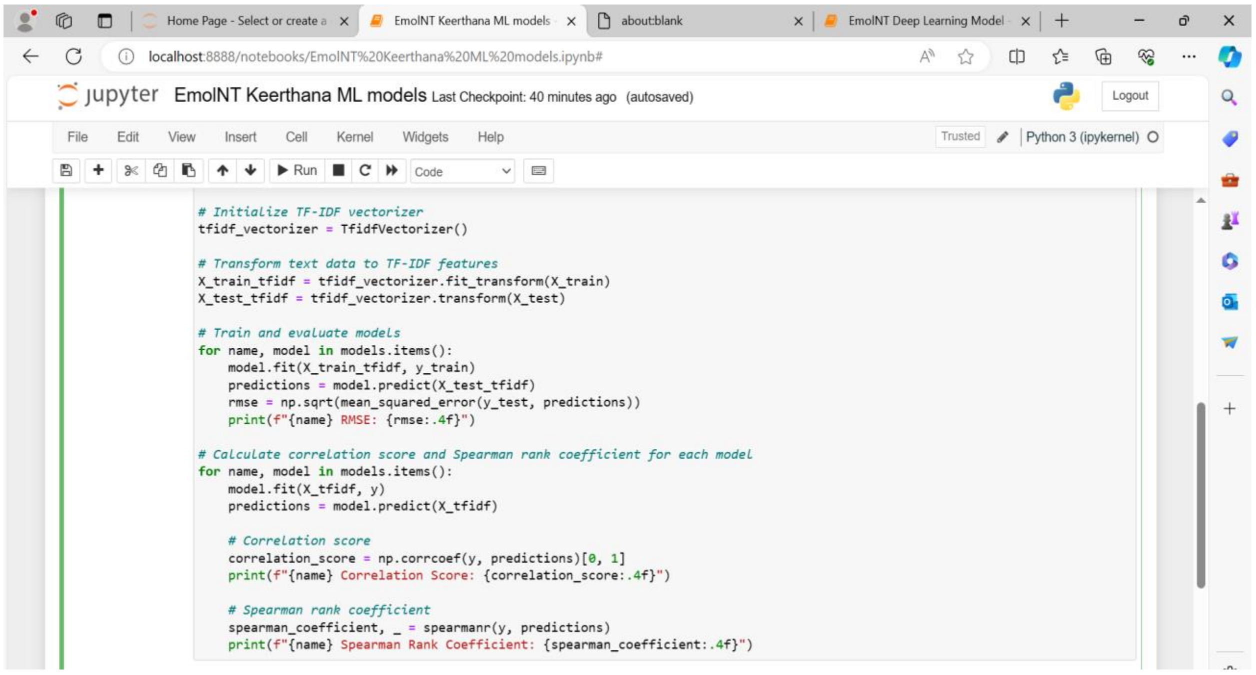
SNAPSHOTS:

CODE FOR DEVELOPING MACHINE LEARNING MODEL: (python code available in github repository)


Home Page - Select or create a
EmoINT Keerthana ML models
about:blank
EmoINT Deep Learning Model
localhost:8888/notebooks/EmoINT%20Keerthana%20ML%20models.ipynb#
jupyter EmoINT Keerthana ML models Last Checkpoint: 38 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [1]:

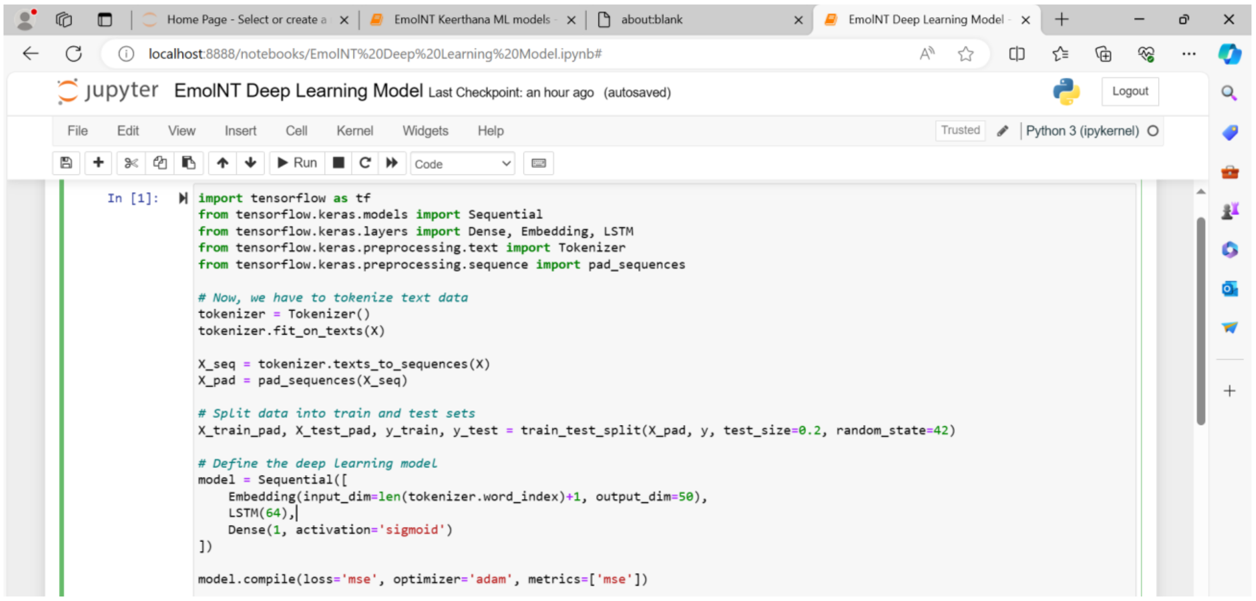
```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import Ridge, LinearRegression
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

# Now I have to Load data from text file
data_url = "https://saifmohammad.com/WebDocs/EmoInt%20Dev%20Data/anger-ratings-0to1.dev.target.txt"
data = pd.read_csv(data_url, header=None, delimiter='\t', names=['Text', 'Intensity'])

# Preprocessing data
X = data['Text']
y = data['Intensity']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize models
models = {
    "Ridge Regression": Ridge(),
    "Linear Regression": LinearRegression(),
    "Support Vector Regression": SVR(),
    "Random Forest Regressor": RandomForestRegressor()
}
```

The following code tokenizes the lexicons and calculates the evaluation metrics:

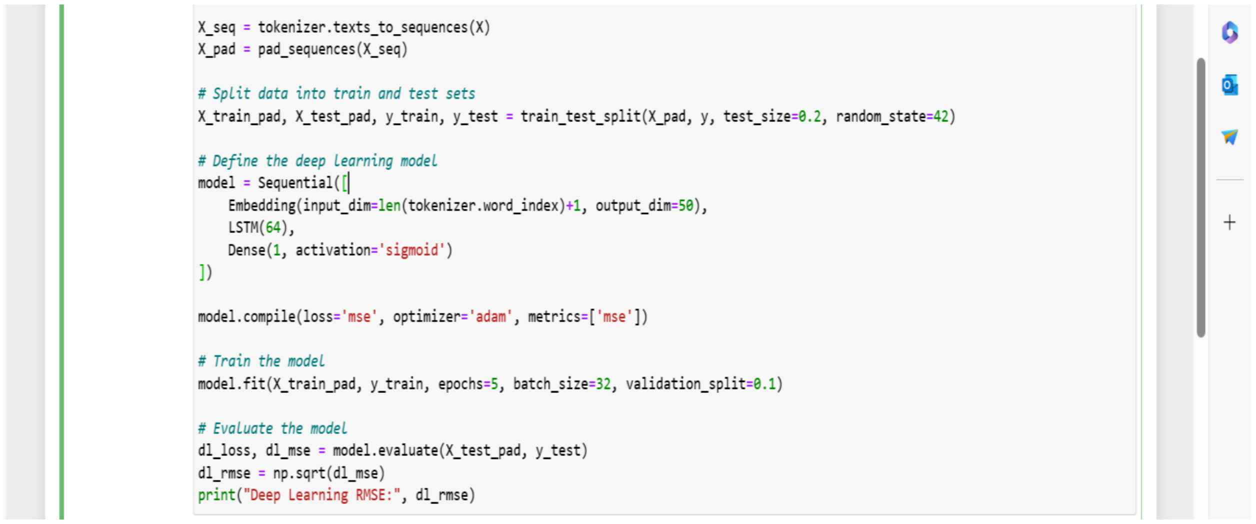

Home Page - Select or create a
EmoINT Keerthana ML models
about:blank
EmoINT Deep Learning Model
localhost:8888/notebooks/EmoINT%20Keerthana%20ML%20models.ipynb#
jupyter EmoINT Keerthana ML models Last Checkpoint: 40 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```python
# Initialize TF-IDF vectorizer
tfidf_vectorizer = TfidfVectorizer()

# Transform text data to TF-IDF features
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Train and evaluate models
for name, model in models.items():
    model.fit(X_train_tfidf, y_train)
    predictions = model.predict(X_test_tfidf)
    rmse = np.sqrt(mean_squared_error(y_test, predictions))
    print(f"{name} RMSE: {rmse:.4f}")

# Calculate correlation score and Spearman rank coefficient for each model
for name, model in models.items():
    model.fit(X_tfidf, y)
    predictions = model.predict(X_tfidf)

    # Correlation score
    correlation_score = np.corrcoef(y, predictions)[0, 1]
    print(f"{name} Correlation Score: {correlation_score:.4f}")

    # Spearman rank coefficient
    spearman_coefficient, _ = spearmanr(y, predictions)
    print(f"{name} Spearman Rank Coefficient: {spearman_coefficient:.4f}")
```

CODE FOR DEVELOPING DEEP LEARNING MODEL:


Home Page - Select or create a
EmoINT Keerthana ML models
about:blank
EmoINT Deep Learning Model
localhost:8888/notebooks/EmoINT%20Deep%20Learning%20Model.ipynb#
jupyter EmoINT Deep Learning Model Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

In [1]:
```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, LSTM
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Now, we have to tokenize text data
tokenizer = Tokenizer()
tokenizer.fit_on_texts(X)

X_seq = tokenizer.texts_to_sequences(X)
X_pad = pad_sequences(X_seq)

# Split data into train and test sets
X_train_pad, X_test_pad, y_train, y_test = train_test_split(X_pad, y, test_size=0.2, random_state=42)

# Define the deep learning model
model = Sequential([
    Embedding(input_dim=len(tokenizer.word_index)+1, output_dim=50),
    LSTM(64),
    Dense(1, activation='sigmoid')
])

model.compile(loss='mse', optimizer='adam', metrics=['mse'])
```



```python
X_seq = tokenizer.texts_to_sequences(X)
X_pad = pad_sequences(X_seq)

# Split data into train and test sets
X_train_pad, X_test_pad, y_train, y_test = train_test_split(X_pad, y, test_size=0.2, random_state=42)

# Define the deep learning model
model = Sequential([
    Embedding(input_dim=len(tokenizer.word_index)+1, output_dim=50),
    LSTM(64),
    Dense(1, activation='sigmoid')
])

model.compile(loss='mse', optimizer='adam', metrics=['mse'])

# Train the model
model.fit(X_train_pad, y_train, epochs=5, batch_size=32, validation_split=0.1)

# Evaluate the model
dl_loss, dl_mse = model.evaluate(X_test_pad, y_test)
dl_rmse = np.sqrt(dl_mse)
print("Deep Learning RMSE:", dl_rmse)
```

OBSERVATIONS:

In my opinion after thorough research, according to me, the best algorithm for emotion intensity identification depends on various factors such as the size and characteristics of the dataset, the complexity of the emotion intensity prediction task, and the computational resources available. In this task I have experimented with multiple algorithms and selected the one that yields the best performance based on evaluation metrics such as RMSE or correlation coefficients. The observations based on the algorithms that I have used to develop the machine learning model is as follows:

1. **Ridge Regression**:
    - Regularized linear regression model that penalizes large coefficients to prevent overfitting.
    - Can perform well when the relationship between features and target variable is linear.
2. **Linear Regression**:
    - Simple and interpretable regression model that assumes a linear relationship between features and target variable.
    - Prone to overfitting if the dataset contains noisy or irrelevant features.
    - May perform well when the relationship between features and target variable is approximately linear and the dataset is not too complex.
3. **Support Vector Regression (SVR)**:
    - Regression model based on support vector machines (SVMs) that aims to find a hyperplane in a high-dimensional space that has the maximum margin from the data points.
4. **Random Forest Regressor**:
    - Ensemble learning method that builds multiple decision trees and averages their predictions to improve generalization and reduce overfitting.

From the output, **Random Forest Regressor has the highest accuracy score**.

Additionally, deep learning approaches such as recurrent neural networks (RNNs) or long short-term memory networks (LSTMs) have shown promising results in emotion intensity prediction.

CONCLUSION:

Ultimately both machine learning and deep learning methods hold potential, for tasks involving identifying the intensity of emotions.

Machine learning techniques like Ridge Regression, Linear Regression, Support Vector Regression and Random Forest Regressor offer simplicity, interpretability and effectiveness in capturing relationships between text features and emotion intensity.

On the side deep learning models, recurrent neural networks (RNNs) and long short term memory networks (LSTMs) excel at detecting intricate sequential patterns in text data. This ability makes them suitable, for tasks related to determining emotion intensity. Deep learning models have the capability to automatically learn representations of text features potentially resulting in performance—especially in scenarios involving extensive datasets or complex emotional nuances.