SENTIMENT ANALYSIS USING CRISP-DM

Leveraging the CRISP-DM methodology to perform sentiment analysis on tweets and build a real-time predictive interface.

UNDERSTANDING AND PREPARING THE DATA

Load Datasets

Load three datasets containing tweets with different sentiments: negative, neutral, and positive.

Use Pandas

Use Pandas to read the CSV files into DataFrames and handle special characters by specifying the encoding format as ISO-8859-1.

Inspect Data

Inspect the initial view of the data before any preprocessing or analysis to understand the characteristics of the datasets.

By loading the datasets, using Pandas to read the CSV files, and specifying the encoding format, we can prepare the data for further sentiment analysis.

LABELING THE DATA

- ## Assign Sentiment Labels

  Assign sentiment labels (negative, neutral, positive) to the tweets in the respective DataFrames.

- ## Combine DataFrames

  Combine the labeled DataFrames into a single DataFrame, keeping only the text and sentiment columns.

- ## Shuffle and Reset Index

  Randomly shuffle the combined DataFrame and reset the index to ensure the data is in a random order.

- ## Visualize Sentiment Distribution

  Plot the distribution of sentiments using Seaborn's countplot to understand the balance of the dataset.

INITIAL DATA SPLIT

Step 1

Split the combined dataset into 80% Training Data and 20% Temporary Data (temp_data)

Step 2

Further split the temp_data into 10% Validation Data and 10% Test Data

Step 3

Ensure the sentiment distribution is balanced across the Training, Validation, and Test sets using stratification

Step 4

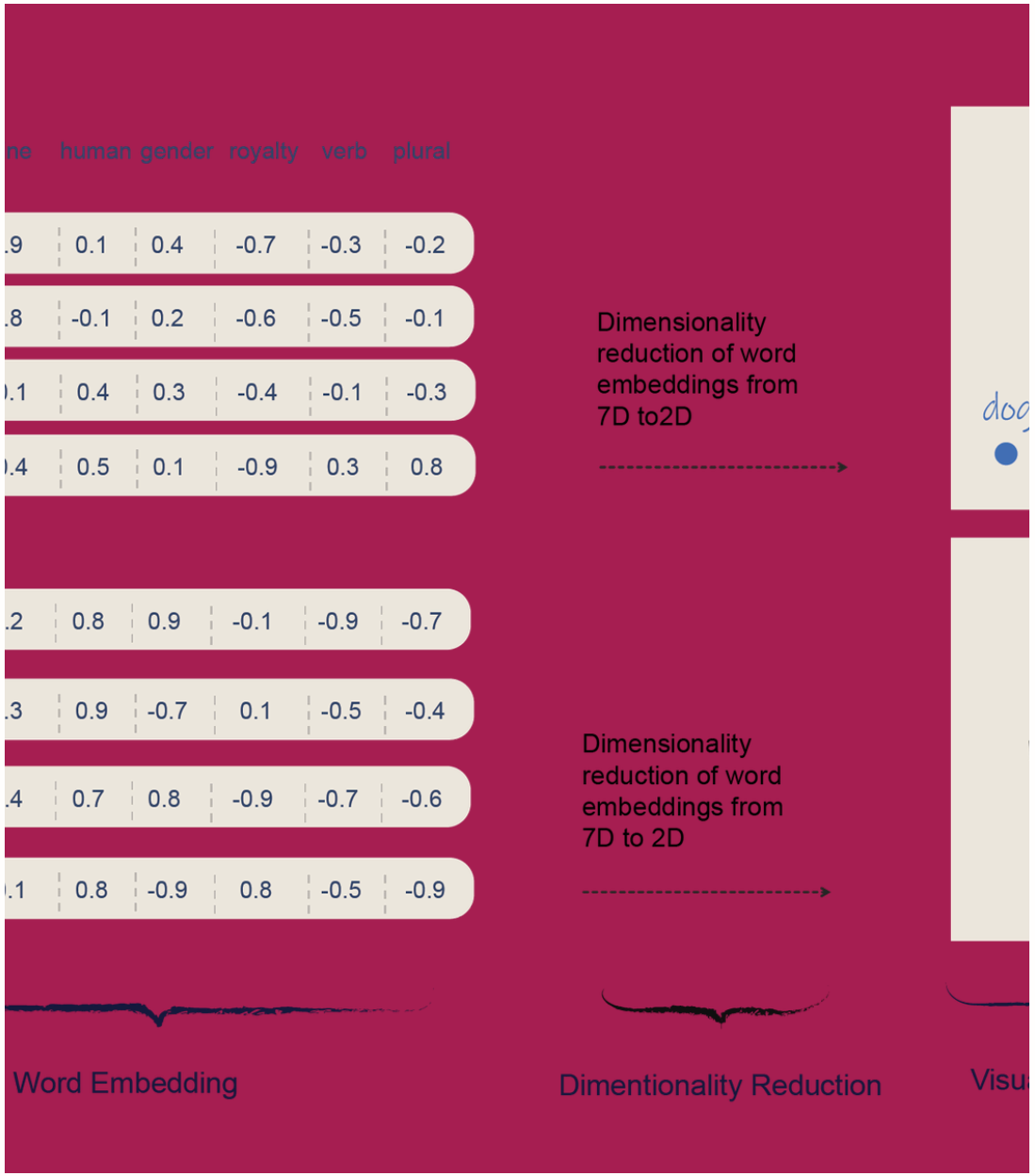
Set a random state (random_state=42) to ensure the split is reproducible

Step 5

Create TensorFlow datasets for Training, Validation, and Test sets to feed into the model

VECTORIZING TEXT DATA

In this section, we focus on converting the textual tweet data into a numerical format that can be fed into a machine learning model. We use the TextVectorization layer from TensorFlow to handle this process. The TextVectorization layer maps the text data into a sequence of numerical token indices, with each token representing a unique word in the vocabulary. This numerical representation allows the model to understand and process the text data effectively.



PREPARING TENSORFLOW DATASETS

Creating Training Dataset

Created a TensorFlow dataset from the training data using tf.data.Dataset.from_tensor_slices(). This dataset contains the text data and corresponding sentiment labels.

Creating Validation Dataset

Created a TensorFlow dataset from the validation data using tf.data.Dataset.from_tensor_slices(). This dataset is used to monitor model performance during training.

Creating Test Dataset

Created a TensorFlow dataset from the test data using tf.data.Dataset.from_tensor_slices(). This dataset is used to evaluate the final model performance.

Batching the Datasets

Batched each dataset in groups of 32 samples for efficient training using the batch() method. This allows the model to learn from the data in small, manageable chunks.

Preparing for Model Input

The TensorFlow datasets now contain text data and corresponding sentiment labels in the correct format for feeding into the machine learning model.

MODEL ARCHITECTURE

MODEL TRAINING

Training the Model

Use the training dataset (80% of the total data) to train the model. The model learns the patterns and relationships in the text data and sentiment labels during this phase.

Monitoring Validation Performance

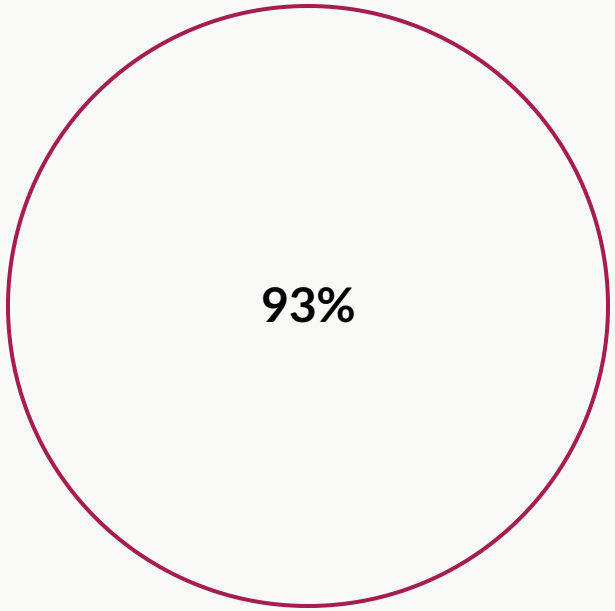
Use the validation dataset (10% of the total data) to monitor the model's performance during training. The validation set is not used for training, but rather to evaluate the model's generalization capabilities and guide hyperparameter tuning.

Tensorboard Visualization

Use callbacks to log the training progress, including metrics like loss and accuracy, for both the training and validation sets. This allows you to visualize the model's performance over time using TensorBoard, a tool for exploring, visualizing, and understanding your machine learning models.

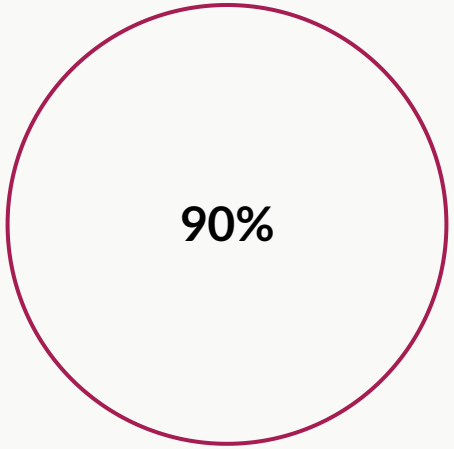
MODEL EVALUATION

Accuracy, Precision, Recall, and F1-Score for each sentiment class



Negative Sentiment

Neutral Sentiment



Positive Sentiment