**Step-by-Step Guide to Evaluating Text-Based AI Models**

When developing AI models that generate text, it's crucial to assess their accuracy and quality through a set of standardized metrics. This guide demonstrates how to evaluate a model using classification metrics and text similarity scores.

**Part 1: Setting Up the Evaluation**

**Libraries and Data**

First, we import necessary Python libraries and prepare the true and generated answers:

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, fbeta\_score

from nltk.translate import sentence\_bleu

from collections import defaultdict

# Example dataset of true answers and model-generated responses

true\_answers = {

"deadline for annual benefits enrollment": "October 20, 2021",

"HSA contribution limit": "$3,650 for employee-only coverage or $7,300 for family coverage",

"medical plan choices": "Consumer Directed Plan and Consumer Directed High Deductible Plan"

}

generated\_answers = {

"deadline for annual benefits enrollment": "October 20, 2021",

"HSA contribution limit": "3650 dollars for an individual",

"medical plan choices": "Directed Consumer Plan and High Deductible Plan"}

**Part 2: Binary Conversion for Evaluation**

**Classification Metrics Preparation**

We convert answers to binary labels to prepare for accuracy and other classification metrics:

# Convert answers to binary: 1 for correct, 0 for incorrect

true\_binary = [1, 1, 1] # Assume all true answers are correct

generated\_binary = [1 if generated\_answers[key] == true\_answers[key] else 0 for key in true\_answers]

**Part 3: Metric Calculation Functions**

**Defining Functions for Metrics**

We define functions to calculate basic metrics and F-beta scores:

def calculate\_basic\_metrics(true\_labels, predicted\_labels):

# Calculates accuracy, precision, recall, and F1 score

return {

'Accuracy': accuracy\_score(true\_labels, predicted\_labels),

'Precision': precision\_score(true\_labels, predicted\_labels),

'Recall': recall\_score(true\_labels, predicted\_labels),

'F1 Score': f1\_score(true\_labels, predicted\_labels)

}

def calculate\_fbeta\_scores(true\_labels, predicted\_labels):

# F2 emphasizes recall, F0.5 emphasizes precision

return {

'F2 Score': fbeta\_score(true\_labels, predicted\_labels, beta=2),

'F0.5 Score': fbeta\_score(true\_labels, predicted\_labels, beta=0.5) }

**Part 4: BLEU Score for Text Quality**

**Evaluating Text Quality**

The BLEU score function evaluates how closely the generated text matches the true text:

def calculate\_text\_metrics(true\_answers, generated\_answers):

bleu\_scores = defaultdict(float)

for key, true\_sentence in true\_answers.items():

reference = [true\_sentence.split()]

hypothesis = generated\_answers[key].split()

bleu\_scores[key] = sentence\_bleu([reference], hypothesis)

return bleu\_scores

**Part 5: Execute and Interpret Results**

**Calculating and Displaying Results**

Finally, we calculate and print the metrics:

# Calculate all metrics

basic\_metrics = calculate\_basic\_metrics(true\_binary, generated\_binary)

fbeta\_metrics = calculate\_fbeta\_scores(true\_binary, generated\_binary)

bleu\_scores = calculate\_text\_metrics(true\_answers, generated\_answers)

# Output the results

print("Basic Metrics:", basic\_metrics)

print("F-Beta Metrics:", fbeta\_metrics)

print("BLEU Scores:", bleu\_scores)

**Interpreting the Results:**

- Basic Metrics: Show the overall accuracy, precision, recall, and F1 score of the model based on binary correctness.

- F-Beta Scores: Highlight the balance between precision and recall, with F2 favoring recall and F0.5 favoring precision, useful in scenarios where one is more important than the other.

- BLEU Scores: Provide a linguistic quality measure based on how similar the generated text is to the reference, ideal for text generation tasks.

Understanding these metrics helps fine-tune AI models to better perform in real-world tasks, ensuring that the generated text meets the desired standards of accuracy and quality.

**Code:**

# Importing necessary libraries for evaluation metrics

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, fbeta\_score

from nltk.translate.bleu\_score import sentence\_bleu

from collections import defaultdict

import numpy as np

# True answers provided for evaluation

true\_answers = {

"deadline for annual benefits enrollment": "October 20, 2021",

"HSA contribution limit": "$3,650 for employee-only coverage or $7,300 for family coverage",

"medical plan choices": "Consumer Directed Plan and Consumer Directed High Deductible Plan"

}

# Model's generated answers to be evaluated against true answers

generated\_answers = {

"deadline for annual benefits enrollment": "October 20, 2021",

"HSA contribution limit": "3650 dollars for an individual",

"medical plan choices": "Directed Consumer Plan and High Deductible Plan"

}

# Convert the true and generated answers into binary labels for classification metrics

# 1 for a correct answer, 0 for an incorrect answer

true\_binary = [1, 1, 1] # All true answers are assumed correct for this example

generated\_binary = [1 if generated\_answers[key] == true\_answers[key] else 0 for key in true\_answers]

# Function to calculate basic classification metrics

def calculate\_basic\_metrics(true\_labels, predicted\_labels):

# Calculates accuracy, precision, recall, and F1 score

# These are common metrics for binary classification tasks

return {

'Accuracy': accuracy\_score(true\_labels, predicted\_labels),

'Precision': precision\_score(true\_labels, predicted\_labels),

'Recall': recall\_score(true\_labels, predicted\_labels),

'F1 Score': f1\_score(true\_labels, predicted\_labels)

}

# Function to calculate F-beta scores which are variations of F1 score

def calculate\_fbeta\_scores(true\_labels, predicted\_labels):

# F2 score puts more emphasis on recall

# F0.5 score puts more emphasis on precision

return {

'F2 Score': fbeta\_score(true\_labels, predicted\_labels, beta=2),

'F0.5 Score': fbeta\_score(true\_labels, predicted\_labels, beta=0.5)

}

# Function to calculate BLEU score for text evaluation

def calculate\_text\_metrics(true\_answers, generated\_answers):

# Stores the BLEU scores for each question-answer pair

bleu\_scores = defaultdict(float)

# Loop through each key-value pair in the true answers

for key, true\_sentence in true\_answers.items():

# Split the sentences into words for BLEU score calculation

reference = [true\_sentence.split()]

hypothesis = generated\_answers[key].split()

# Calculate the BLEU score and store it in the dictionary

bleu\_scores[key] = sentence\_bleu(reference, hypothesis)

# Return the dictionary of BLEU scores

return bleu\_scores

# Calculate the basic metrics using the functions defined above

basic\_metrics = calculate\_basic\_metrics(true\_binary, generated\_binary)

# Calculate the F-beta scores

fbeta\_metrics = calculate\_fbeta\_scores(true\_binary, generated\_binary)

# Calculate the BLEU scores for text quality

bleu\_scores = calculate\_text\_metrics(true\_answers, generated\_answers)

# Output the results of the metrics calculations

print("Basic Metrics:", basic\_metrics)

print("F-Beta Metrics:", fbeta\_metrics)

print("BLEU Scores:", bleu\_scores)

**Example Output:**

