Ex2: Bias Audit of Pre-Trained Word Embeddings Using the Word Embedding Association Test (WEAT)

Learning Objective:
Develop a program that performs a bias audit using Word Embedding Association Test (WEAT), to identify and quantify gender or racial stereotypes present in Google News Word2Vec embeddings.

Steps:

1. Import necessary Python libraries: gensim, numpy, etc.
2. Load the Google News Word2Vec embeddings. (2GB Dataset, takes time)
3. Define Target and Attribute Word Sets
    a. Target words (X and Y): Groups to compare, e.g.,
        i. Gender: `X = ["man","male","boy"……],`
        `Y = ["woman","female","girl"……]`
    b. Attribute words (A and B): Concepts to test association with, e.g.,
        i. Career vs Family: `A = ["career","business"……],`
        `B = ["family","home"……]`
4. Define Helper Functions – association function, weat score function, effect size function and compute weat score, effect size.
5. Interpret the Results as
    a. Effect Size > 0: Target group X is more associated with A (bias towards X-A).
    b. Effect Size < 0: Target group Y is more associated with A (bias towards Y-A).
    c. `Effect Size ≈ 0: No detectable bias.`

Optional Extensions

1. Perform multiple WEAT tests for different bias types (**gender, racial, religious**).
2. Visualize effect sizes using bar charts for comparison.

Program:

```
import gensim.downloader as api
import numpy as np
import random
print("Loading pretrained model...")
model = api.load("glove-wiki-gigaword-100")
print("Model Loaded\n")

X = ["career", "salary", "office", "management"]
Y = ["home", "family", "children", "marriage"]

A = ["man", "male", "he"]
B = ["woman", "female", "she"]

X = [w for w in X if w in model]
Y = [w for w in Y if w in model]
A = [w for w in A if w in model]
```

```
B = [w for w in B if w in model]

def assoc(w):
    return np.mean([model.similarity(w, a) for a in A]) -
\
           np.mean([model.similarity(w, b) for b in B])

weat_score = sum(assoc(x) for x in X) - sum(assoc(y) for
y in Y)

combined = X + Y
mean_X = np.mean([assoc(x) for x in X])
mean_Y = np.mean([assoc(y) for y in Y])
std_dev = np.std([assoc(w) for w in combined])

effect_size = (mean_X - mean_Y) / std_dev

def permutation_test(iters=500):
    observed = weat_score
    combined = X + Y
    size = len(X)
    count = 0

    for _ in range(iters):
        random.shuffle(combined)
        X_i = combined[:size]
        Y_i = combined[size:]
        score = sum(assoc(x) for x in X_i) - sum(assoc(y)
for y in Y_i)
        if score > observed:
            count += 1
    return count / iters

p_value = permutation_test()

print("WEAT Score  :", round(weat_score, 4))
print("Effect Size :", round(effect_size, 4))
print("P-Value     :", round(p_value, 4))

if p_value < 0.05:
    print("Bias Detected")
else:
```

```
    print("No Significant Bias")
```

Output:

```
Loading pretrained model...
Model Loaded

WEAT Score  : 0.3091
Effect Size : 1.3601
P-Value     : 0.03
Bias Detected
```

Learning Outcome:
Students will be able to process word embedding datasets, WEAT was implemented to measure associations between target and attribute word sets, interpreted effect sizes to detect bias, and critically evaluated the presence of gender or racial stereotypes in pre-trained embeddings.