In [ ]:
```
import torch
from sentence_transformers import SentenceTransformer
from sklearn.metrics.pairwise import cosine_similarity
import pandas as pd
from tqdm import tqdm
```

In [ ]:
```
# Set the device
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# Load the model
model = SentenceTransformer('pritamdeka/BioBERT-mnli-snli-scinli-scitail-
```

In [20]:
```
df = pd.read_csv('AthalianaOrthologer2_Hayai_annotation_v3.2.tsv', sep='\
df = df[['Product_Name', 'Zen_OrthoDB_Desc']]
```

In [ ]:
```
# Handle missing values
df.dropna(subset=['Product_Name', 'Zen_OrthoDB_Desc'], inplace=True)

# Remove rows where 'Product_Name' or 'Zen_OrthoDB_Desc' contain "unchara
df = df[~df['Product_Name'].str.contains('uncharacterized|hypothetical',
        ~df['Zen_OrthoDB_Desc'].str.contains('uncharacterized|hypothetica

# Remove rows in 'Product_Name' that start with 'At', 'Emb_', 'Gb_'
df = df[~df['Product_Name'].str.contains(r'^At\d+', case=False, na=False)
        ~df['Zen_OrthoDB_Desc'].str.contains(r'^At\d+', case=False, na=Fa

df = df[~df['Product_Name'].str.contains(r'^Emb_|^Gb_', case=False, na=Fa
        ~df['Zen_OrthoDB_Desc'].str.contains(r'^Emb_|^Gb_', case=False, n
```

In [28]:
```
# Convert descriptions to strings
df['Product_Name'] = df['Product_Name'].astype(str)
df['Zen_OrthoDB_Desc'] = df['Zen_OrthoDB_Desc'].astype(str)

# Extract the descriptions
descriptions_1 = df['Product_Name'].tolist()
descriptions_2 = df['Zen_OrthoDB_Desc'].tolist()

# Initialize an empty list to store similarity scores
similarity_scores = []
```

In [ ]:
```
# Process in batches
batch_size = 32
num_batches = (len(descriptions_1) + batch_size - 1) // batch_size

for batch_num in tqdm(range(num_batches)):
    start_idx = batch_num * batch_size
    end_idx = min(start_idx + batch_size, len(descriptions_1))
    batch_descriptions_1 = descriptions_1[start_idx:end_idx]
    batch_descriptions_2 = descriptions_2[start_idx:end_idx]

    # Generate embeddings
    embeddings_1 = model.encode(batch_descriptions_1, device=device, show
    embeddings_2 = model.encode(batch_descriptions_2, device=device, show

    # Compute cosine similarity for each pair in the batch
    batch_similarity_scores = [
        float(cosine_similarity([embeddings_1[i]], [embeddings_2[i]])[0][
    ]

    similarity_scores.extend(batch_similarity_scores)

# Add similarity scores to the DataFrame
df['similarity_score'] = similarity_scores
```

In [30]:

```
# Save the DataFrame with similarity scores
df.to_csv('protein_similarity_scores.tsv', sep='\t', index=False)

# Display the first few results
print(df.head())
```

```
                                        Product_Name  \
0                             Rho termination factor
1  1-aminocyclopropane-1-carboxylate oxidase homo...
3  Bifunctional inhibitor/lipid-transfer protein/...
4  DENN domain and WD repeat-containing protein SCD1
5                               F-box protein SKIP24

                                    Zen_OrthoDB_Desc  similarity_score
0                  Rho-N domain-containing protein 1          0.429524
1  1-aminocyclopropane-1-carboxylate oxidase homo...          1.000000
3  Bifunctional inhibitor/plant lipid transfer pr...          0.879570
4  DENN domain and WD repeat-containing protein SCD1          1.000000
5                               F-box protein SKIP24          1.000000
```

In [31]:

```
mean_score = df['similarity_score'].mean()
median_score = df['similarity_score'].median()
std_score = df['similarity_score'].std()
min_score = df['similarity_score'].min()
max_score = df['similarity_score'].max()

print(f"Mean similarity score: {mean_score:.4f}")
print(f"Median similarity score: {median_score:.4f}")
print(f"Standard deviation: {std_score:.4f}")
print(f"Minimum score: {min_score:.4f}")
print(f"Maximum score: {max_score:.4f}")
```

```
Mean similarity score: 0.7104
Median similarity score: 0.7777
Standard deviation: 0.2763
Minimum score: -0.0723
Maximum score: 1.0000
```

In [32]:

```
threshold = 0.5
df['is_similar'] = df['similarity_score'] >= threshold
num_similar = df['is_similar'].sum()
total_pairs = len(df)
similar_percentage = (num_similar / total_pairs) * 100

print(f"Number of similar pairs: {num_similar}")
print(f"Total pairs: {total_pairs}")
print(f"Percentage of similar pairs: {similar_percentage:.2f}%")
```

```
Number of similar pairs: 15617
Total pairs: 20490
Percentage of similar pairs: 76.22%
```

In [33]:

```
# High similarity samples
high_similarity_samples = df[df['similarity_score'] >= threshold].sample(
```

```python
print("High Similarity Samples:")
print(high_similarity_samples[['Product_Name', 'Zen_OrthoDB_Desc', 'simil
```

```
High Similarity Samples:
                                          Product_Name  \
4636                               Protein IQ-DOMAIN 7   
23271                                 Zinc transporter   
23464      Respiratory burst oxidase homolog protein C   
1852   Ethylene-responsive transcription factor ERF118   
23616             F-box/kelch-repeat protein At5g15710   

                                      Zen_OrthoDB_Desc  similarity_score  
4636                    protein IQ-DOMAIN 1 isoform X1          0.549217  
23271                               zinc transporter 5          0.775821  
23464      respiratory burst oxidase homolog protein C          1.000000  
1852   Ethylene-responsive transcription factor ERF118          1.000000  
23616                       F-box/kelch-repeat protein          0.831193  
```

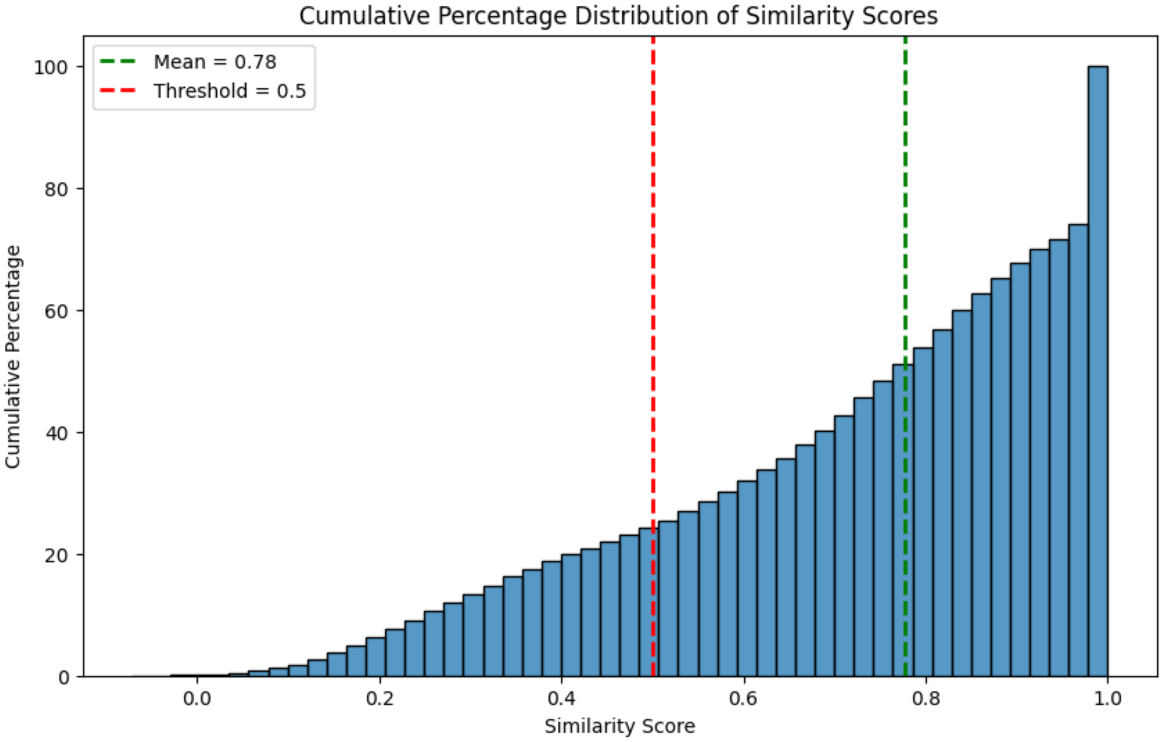
In [34]:

```python
# Low similarity samples
low_similarity_samples = df[df['similarity_score'] < threshold].sample(15
print("\nLow Similarity Samples:")
print(low_similarity_samples[['Product_Name', 'Zen_OrthoDB_Desc', 'simila
```

```
Low Similarity Samples:
                                            Product_Name  \
20413          Late embryogenesis abundant (LEA) protein   
864          7-dehydrocholesterol reductase-like protein   
11113                          Defensin-like protein 196   
18898  Calcium-dependent lipid-binding (CaLB domain) ...   
2604            Toll/interleukin-1 receptor-like protein   
5949                          Ras-related protein RABA5e   
17288                DDB1- and CUL4-associated factor 13   
789             Germin-like protein subfamily T member 3   
10535                                       Protein NPG1   
7190                      Putative reverse transcriptase   
9548    Cysteine/Histidine-rich C1 domain family protein   
27165                 PROTEIN TARGETING TO STARCH (PTST)   
25309  2-(3-amino-3-carboxypropyl)histidine synthase ...   
6213            Leucine-rich repeat (LRR) family protein   
11461                          Transcription factor PRE5   

                                        Zen_OrthoDB_Desc  similarity_score  
20413                             embryonic protein DC-8          0.496653  
864                               Zinc finger, RING-type          0.358341  
11113                       Knottin, scorpion toxin-like          0.254267  
18898                        Elicitor-responsive protein          0.322947  
2604                          Disease resistance protein          0.231515  
5949                                        Small GTPase          0.322744  
17288                      WD40-repeat-containing domain          0.263313  
789                   RmlC-like cupin domain superfamily          0.338557  
10535                Large ribosomal subunit protein uL3          0.243686  
7190              Ribonuclease H-like domain, plant type          0.403012  
9548                               Zinc finger, PHD-type          0.335340  
27165  AMP-activated protein kinase glycogen-binding ...          0.272699  
25309          Diphthamide synthesis DPH2 family protein          0.372453  
6213                                              kinase          0.156915  
11461     Myc-type, basic helix-loop-helix (bHLH) domain          0.199195  
```

In [55]:

```python
plt.figure(figsize=(10, 6))
sns.histplot(df['similarity_score'], bins=50, cumulative=True, stat="perc
plt.title('Cumulative Percentage Distribution of Similarity Scores')
plt.xlabel('Similarity Score')
plt.ylabel('Cumulative Percentage')

mean_score = df['similarity_score'].mean()
median_score = df['similarity_score'].median()
threshold = 0.5

plt.axvline(median_score, color='green', linestyle='dashed', linewidth=2,
plt.axvline(threshold, color='red', linestyle='dashed', linewidth=2, labe

plt.legend()

# Save the plot
plt.savefig('similarity_score_distribution.png', dpi=300, bbox_inches='ti

plt.show()
```


Cumulative Percentage Distribution of Similarity Scores
Mean = 0.78
Threshold = 0.5
Cumulative Percentage
Similarity Score