Pairwise Embedding Distance

One way to measure the similarity (or dissimilarity) between two predictions on a shared or similar input is to embed the predictions and compute a vector distance between the two embeddings.[1]

You can load the `pairwise_embedding_distance` evaluator to do this.

Note: This returns a **distance** score, meaning that the lower the number, the **more** similar the outputs are, according to their embedded representation.

Check out the reference docs for the PairwiseEmbeddingDistanceEvalChain for more info.

```
from langchain.evaluation import load_evaluator

evaluator = load_evaluator("pairwise_embedding_distance")
```

API Reference:

- load_evaluator from `langchain.evaluation`

```
evaluator.evaluate_string_pairs(
    prediction="Seattle is hot in June", prediction_b="Seattle is cool
in June."
)
```

```
    {'score': 0.0966466944859925}
```

```
evaluator.evaluate_string_pairs(
    prediction="Seattle is warm in June", prediction_b="Seattle is cool
in June."
)
```

```
    {'score': 0.03761174337464557}
```

Select the Distance Metric

By default, the evalutor uses cosine distance. You can choose a different distance metric if you'd like.

```
from langchain.evaluation import EmbeddingDistance

list(EmbeddingDistance)
```

API Reference:

- EmbeddingDistance from `langchain.evaluation`

```
    [<EmbeddingDistance.COSINE: 'cosine'>,
     <EmbeddingDistance.EUCLIDEAN: 'euclidean'>,
     <EmbeddingDistance.MANHATTAN: 'manhattan'>,
     <EmbeddingDistance.CHEBYSHEV: 'chebyshev'>,
     <EmbeddingDistance.HAMMING: 'hamming'>]
```

```
evaluator = load_evaluator(
    "pairwise_embedding_distance",
distance_metric=EmbeddingDistance.EUCLIDEAN
)
```

Select Embeddings to Use

The constructor uses `OpenAI` embeddings by default, but you can configure this however you want. Below, use huggingface local embeddings

```
from langchain.embeddings import HuggingFaceEmbeddings

embedding_model = HuggingFaceEmbeddings()
hf_evaluator = load_evaluator("pairwise_embedding_distance",
embeddings=embedding_model)
```

API Reference:

- HuggingFaceEmbeddings from `langchain.embeddings`

```
hf_evaluator.evaluate_string_pairs(
    prediction="Seattle is hot in June", prediction_b="Seattle is cool
in June."
)
```

```
    {'score': 0.5486443280477362}
```

```
hf_evaluator.evaluate_string_pairs(
    prediction="Seattle is warm in June", prediction_b="Seattle is cool
in June."
)
```

```
    {'score': 0.21018880025138598}
```

1. Note: When it comes to semantic similarity, this often gives better results than older string distance metrics (such as those in the `PairwiseStringDistanceEvalChain`), though it tends to be less reliable than evaluators that use the LLM directly (such as the `PairwiseStringEvalChain`)