Embedding Distance

To measure semantic similarity (or dissimilarity) between a prediction and a reference label string, you could use a vector vector distance metric the two embedded representations using the `embedding_distance` evaluator.[1]

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

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

```
from langchain.evaluation import load_evaluator

evaluator = load_evaluator("embedding_distance")
```

API Reference:

- load_evaluator from `langchain.evaluation`

```
evaluator.evaluate_strings(prediction="I shall go", reference="I shan't go")
```

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

```
evaluator.evaluate_strings(prediction="I shall go", reference="I will go")
```

```
    {'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'>]
```

```
# You can load by enum or by raw python string
evaluator = load_evaluator(
    "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("embedding_distance",
embeddings=embedding_model)
```

API Reference:

- HuggingFaceEmbeddings from `langchain.embeddings`

```
hf_evaluator.evaluate_strings(prediction="I shall go", reference="I
shan't go")
```

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

```
hf_evaluator.evaluate_strings(prediction="I shall go", reference="I
will go")
```

```
    {'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 [StringDistanceEvalChain] (https://api.python.langchain.com/en/latest/evaluation/langchain.evaluation.string_distance.base.StringDistanceEvalChain.html#langchain.evaluation.string_distance.base.StringDistanceEvalChain)), though it tends to be less reliable than evaluators that use the LLM directly (such as the [QAEvalChain] (https://api.python.langchain.com/en/latest/evaluation/langchain.evaluation.qa.eval_chain.QAEvalChain.html#langchain.evaluation.qa.eval_chain.QAEvalChain) or [LabeledCriteriaEvalChain] (https://api.python.langchain.com/en/latest/evaluation/langchain.evaluation.criteria.eval_chain.LabeledCriteriaEvalChain.html#langchain.evaluation.criteria.eval_chain.LabeledCriteriaEvalChain))