Select by maximal marginal relevance (MMR)

The `MaxMarginalRelevanceExampleSelector` selects examples based on a combination of which examples are most similar to the inputs, while also optimizing for diversity. It does this by finding the examples with the embeddings that have the greatest cosine similarity with the inputs, and then iteratively adding them while penalizing them for closeness to already selected examples.

```
from langchain.prompts.example_selector import (
    MaxMarginalRelevanceExampleSelector,
    SemanticSimilarityExampleSelector,
)
from langchain.vectorstores import FAISS
from langchain.embeddings import OpenAIEmbeddings
from langchain.prompts import FewShotPromptTemplate, PromptTemplate

example_prompt = PromptTemplate(
    input_variables=["input", "output"],
    template="Input: {input}\nOutput: {output}",
)

# These are a lot of examples of a pretend task of creating antonyms.
examples = [
    {"input": "happy", "output": "sad"},
    {"input": "tall", "output": "short"},
    {"input": "energetic", "output": "lethargic"},
    {"input": "sunny", "output": "gloomy"},
    {"input": "windy", "output": "calm"},
]
```

API Reference:

- MaxMarginalRelevanceExampleSelector from `langchain.prompts.example_selector`
- SemanticSimilarityExampleSelector from `langchain.prompts.example_selector`
- FAISS from `langchain.vectorstores`
- OpenAIEmbeddings from `langchain.embeddings`

- FewShotPromptTemplate from `langchain.prompts`
- PromptTemplate from `langchain.prompts`

```
example_selector = MaxMarginalRelevanceExampleSelector.from_examples(
    # This is the list of examples available to select from.
    examples,
    # This is the embedding class used to produce embeddings which are
used to measure semantic similarity.
    OpenAIEmbeddings(),
    # This is the VectorStore class that is used to store the embeddings
and do a similarity search over.
    FAISS,
    # This is the number of examples to produce.
    k=2,
)
mmr_prompt = FewShotPromptTemplate(
    # We provide an ExampleSelector instead of examples.
    example_selector=example_selector,
    example_prompt=example_prompt,
    prefix="Give the antonym of every input",
    suffix="Input: {adjective}\nOutput:",
    input_variables=["adjective"],
)
```

```
# Input is a feeling, so should select the happy/sad example as the first
one
print(mmr_prompt.format(adjective="worried"))
```

```
    Give the antonym of every input

    Input: happy
    Output: sad

    Input: windy
    Output: calm

    Input: worried
    Output:
```

```
# Let's compare this to what we would just get if we went solely off of
similarity,
# by using SemanticSimilarityExampleSelector instead of
MaxMarginalRelevanceExampleSelector.
example_selector = SemanticSimilarityExampleSelector.from_examples(
    # This is the list of examples available to select from.
    examples,
    # This is the embedding class used to produce embeddings which are
used to measure semantic similarity.
    OpenAIEmbeddings(),
    # This is the VectorStore class that is used to store the embeddings
and do a similarity search over.
    FAISS,
    # This is the number of examples to produce.
    k=2,
)
similar_prompt = FewShotPromptTemplate(
    # We provide an ExampleSelector instead of examples.
    example_selector=example_selector,
    example_prompt=example_prompt,
    prefix="Give the antonym of every input",
    suffix="Input: {adjective}\nOutput:",
    input_variables=["adjective"],
)
print(similar_prompt.format(adjective="worried"))
```

```
    Give the antonym of every input

    Input: happy
    Output: sad

    Input: sunny
    Output: gloomy

    Input: worried
    Output:
```