Question Answering

Contents

This notebook covers how to evaluate generic question answering problems. This is a situation where you have an example containing a question and its corresponding ground truth answer, and you want to measure how well the language model does at answering those questions.

Setup

For demonstration purposes, we will just evaluate a simple question answering system that only evaluates the model's internal knowledge. Please see other notebooks for examples where it evaluates how the model does at question answering over data not present in what the model was trained on.

```
from langchain.prompts import PromptTemplate
from langchain.chains import LLMChain
from langchain.llms import OpenAI
```

```
prompt = PromptTemplate(template="Question: {question}\nAnswer:", input_variables=
["question"])
```

```
llm = OpenAI(model_name="text-davinci-003", temperature=0)
```

Skip to main content

Examples

For this purpose, we will just use two simple hardcoded examples, but see other notebooks for tips on how to get and/or generate these examples.

```
examples = [
    {
        "question": "Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?",
        "answer": "11"
    },
    {
        "question": 'Is the following sentence plausible? "Joao Moutinho caught the screen pass in the NFC championship."',
        "answer": "No"
    }
]
```

Predictions

We can now make and inspect the predictions for these questions.

```
predictions = chain.apply(examples)
```

```
predictions
```

```
[{'text': ' 11 tennis balls'},
 {'text': ' No, this sentence is not plausible. Joao Moutinho is a professional soccer player, not an American football player, so it is not likely that he would be catching a screen pass in the NFC championship.'}]
```

Evaluation

We can see that if we tried to just do exact match on the answer answers (`11` and `No`) they would not match what the language model answered. However, semantically the language

Skip to main content

model is correct in both cases. In order to account for this, we can use a language model itself to evaluate the answers.

```
from langchain.evaluation.qa import QAEvalChain
```

```
llm = OpenAI(temperature=0)
eval_chain = QAEvalChain.from_llm(llm)
graded_outputs = eval_chain.evaluate(examples, predictions,
question_key="question", prediction_key="text")
```

```
for i, eg in enumerate(examples):
    print(f"Example {i}:")
    print("Question: " + eg['question'])
    print("Real Answer: " + eg['answer'])
    print("Predicted Answer: " + predictions[i]['text'])
    print("Predicted Grade: " + graded_outputs[i]['text'])
    print()
```

```
Example 0:
Question: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can
has 3 tennis balls. How many tennis balls does he have now?
Real Answer: 11
Predicted Answer:  11 tennis balls
Predicted Grade:  CORRECT

Example 1:
Question: Is the following sentence plausible? "Joao Moutinho caught the screen
pass in the NFC championship."
Real Answer: No
Predicted Answer:  No, this sentence is not plausible. Joao Moutinho is a
professional soccer player, not an American football player, so it is not likely
that he would be catching a screen pass in the NFC championship.
Predicted Grade:  CORRECT
```

Customize Prompt

You can also customize the prompt that is used. Here is an example prompting it using a score from 0 to 10. The custom prompt requires 3 input variables: “query”, “answer” and “result”. Where “query” is the question, “answer” is the ground truth answer, and “result” is the predicted answer.

Skip to main content

```
from langchain.prompts.prompt import PromptTemplate

_PROMPT_TEMPLATE = """You are an expert professor specialized in grading students'
answers to questions.
You are grading the following question:
{query}
Here is the real answer:
{answer}
You are grading the following predicted answer:
{result}
What grade do you give from 0 to 10, where 0 is the lowest (very low similarity)
and 10 is the highest (very high similarity)?
"""

PROMPT = PromptTemplate(input_variables=["query", "answer", "result"],
template=_PROMPT_TEMPLATE)
```

```
evalchain = QAEvalChain.from_llm(llm=llm,prompt=PROMPT)
evalchain.evaluate(examples, predictions, question_key="question",
answer_key="answer", prediction_key="text")
```

Comparing to other evaluation metrics

We can compare the evaluation results we get to other common evaluation metrics. To do this, let's load some evaluation metrics from HuggingFace's `evaluate` package.

```
# Some data munging to get the examples in the right format
for i, eg in enumerate(examples):
    eg['id'] = str(i)
    eg['answers'] = {"text": [eg['answer']], "answer_start": [0]}
    predictions[i]['id'] = str(i)
    predictions[i]['prediction_text'] = predictions[i]['text']

for p in predictions:
    del p['text']

new_examples = examples.copy()
for eg in new_examples:
    del eg ['question']
    del eg['answer']
```

```
from evaluate import load
squad_metric = load("squad")
```

Skip to main content

```
    predictions=predictions,
)
```

```
results
```

```
{'exact_match': 0.0, 'f1': 28.125}
```