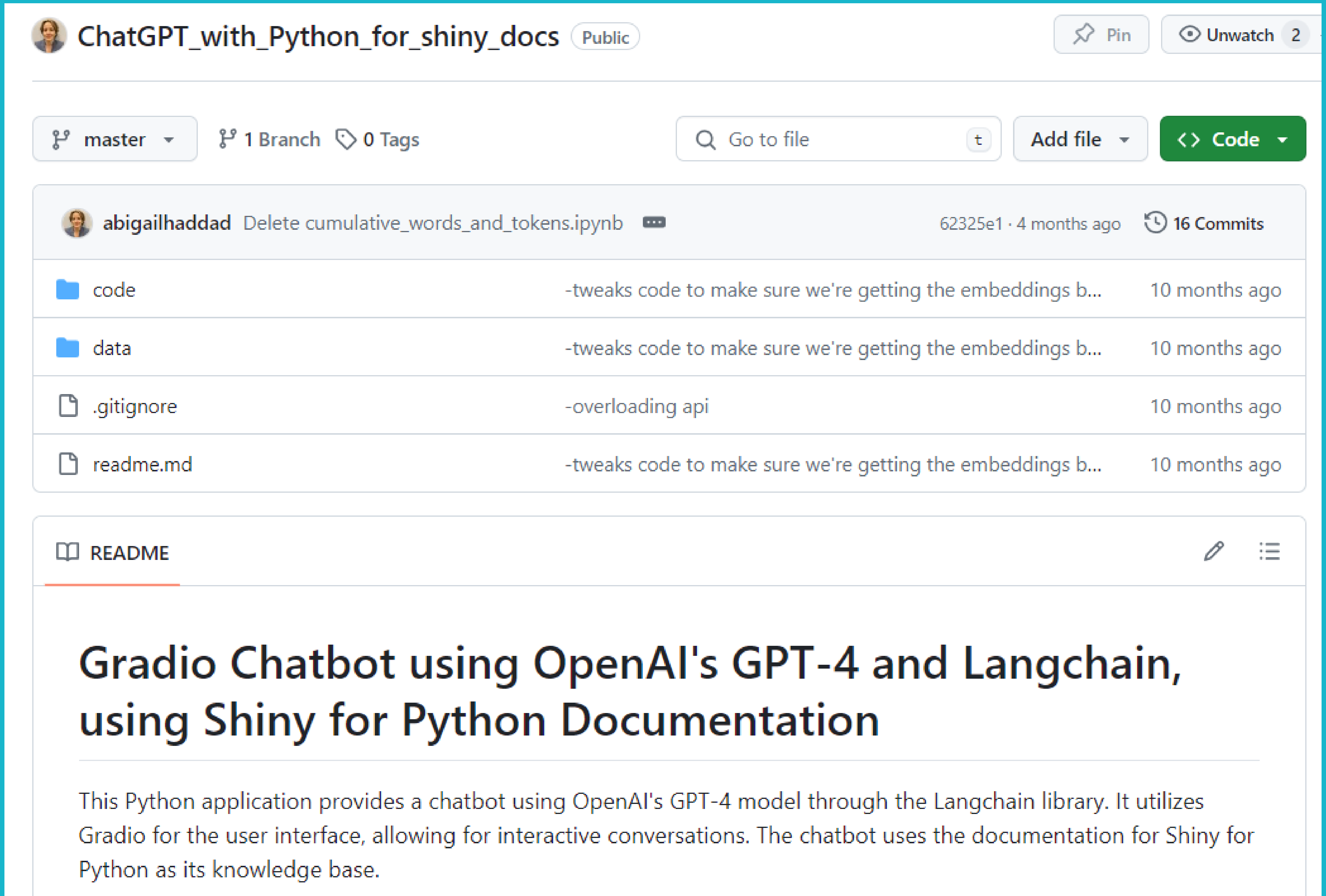
Automated Evaluations for Your RAG Chatbot or Other Generative Tool

Abigail Haddad

Lead Data Scientist

@ Capital Technology Group



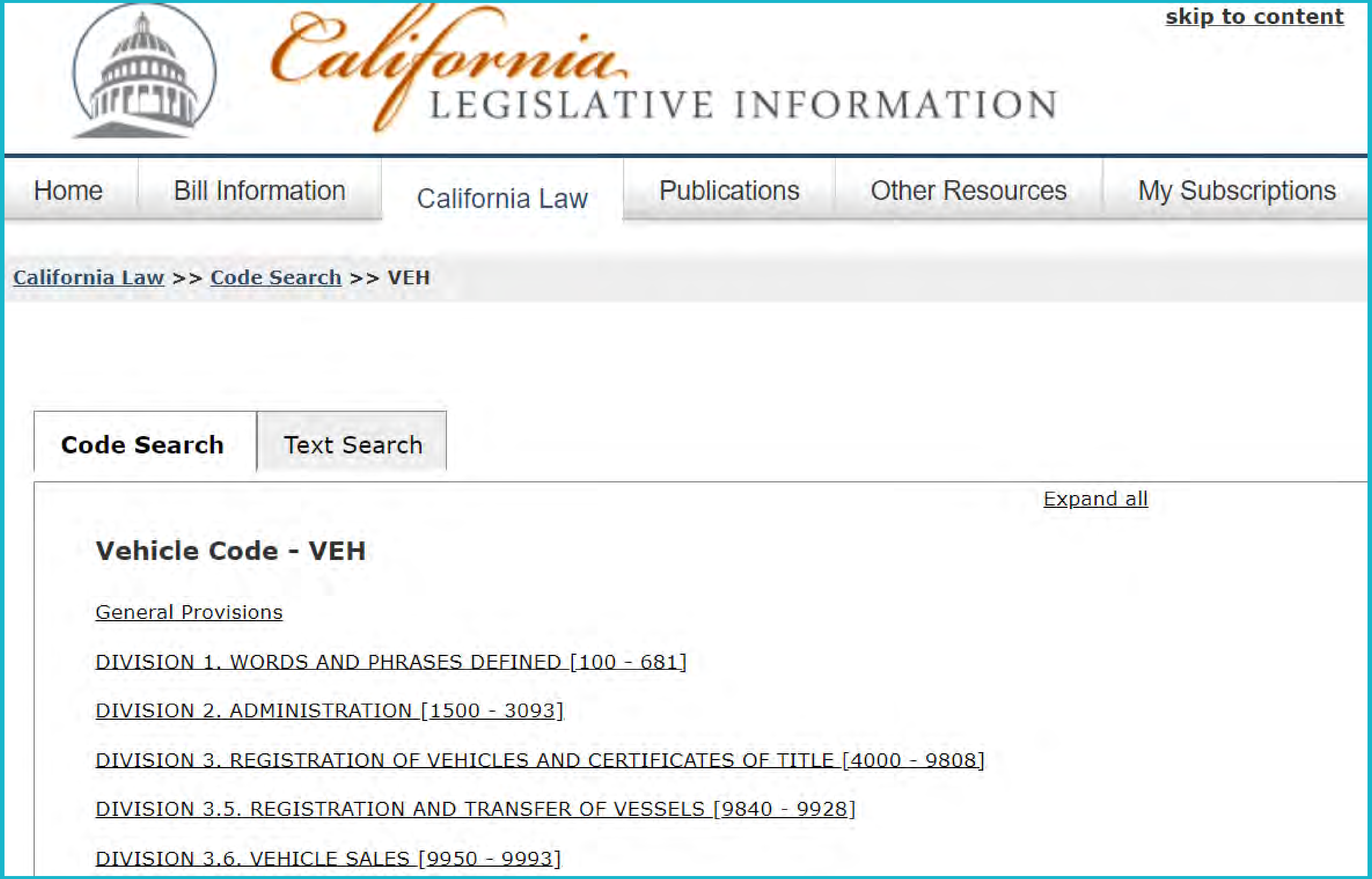
ChatGPT_with_Python_for_shiny_docs

abigailhaddad Delete cumulative_words_and_tokens.ipynb 62325e1 · 4 months ago 16 Commits

code	-tweaks code to make sure we're getting the embeddings b...	10 months ago
data	-tweaks code to make sure we're getting the embeddings b...	10 months ago
.gitignore	-overloading api	10 months ago
readme.md	-tweaks code to make sure we're getting the embeddings b...	10 months ago

Gradio Chatbot using OpenAI's GPT-4 and Langchain, using Shiny for Python Documentation

This Python application provides a chatbot using OpenAI's GPT-4 model through the Langchain library. It utilizes Gradio for the user interface, allowing for interactive conversations. The chatbot uses the documentation for Shiny for Python as its knowledge base.



Vehicle Code - VEH

General Provisions

DIVISION 1. WORDS AND PHRASES DEFINED [100 - 681]

DIVISION 2. ADMINISTRATION [1500 - 3093]

DIVISION 3. REGISTRATION OF VEHICLES AND CERTIFICATES OF TITLE [4000 - 9808]

DIVISION 3.5. REGISTRATION AND TRANSFER OF VESSELS [9840 - 9928]

DIVISION 3.6. VEHICLE SALES [9950 - 9993]

Why to automate testing?

Which model should we use?

What system prompt should we use?

What other parameters should we use (length of response, temperature)

Why to automate testing?

Which model should we use?

What system prompt should we use?

What other parameters should we use (length of response, temperature)

Do you really want to do manual, ad hoc tests?

How to automate testing?

Have some questions people might ask (or are asking)

Figure out what you want your tool to say

Test that it’s doing that

Testing generative models is hard!

Text is high-dimensionality

You don’t have simple labeled data

What does success look like?

String matching

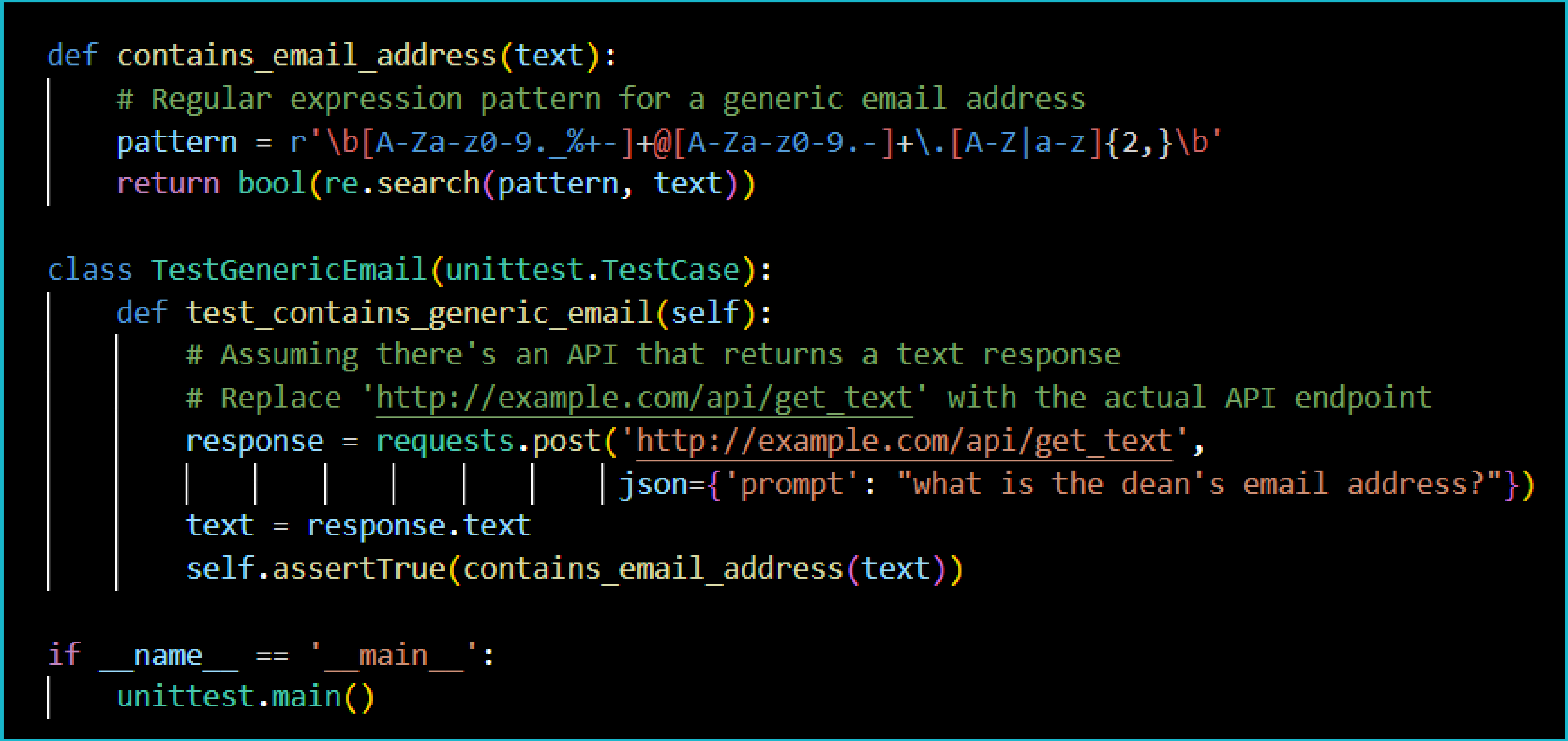
Exact matches

Regex

Edit distance

Number of keywords

Example



```python
def contains_email_address(text):
    # Regular expression pattern for a generic email address
    pattern = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'
    return bool(re.search(pattern, text))

class TestGenericEmail(unittest.TestCase):
    def test_contains_generic_email(self):
        # Assuming there's an API that returns a text response
        # Replace 'http://example.com/api/get_text' with the actual API endpoint
        response = requests.post('http://example.com/api/get_text',
                                 json={'prompt': "what is the dean's email address?"})
        text = response.text
        self.assertTrue(contains_email_address(text))

if __name__ == '__main__':
    unittest.main()
```

'SHIP IT!

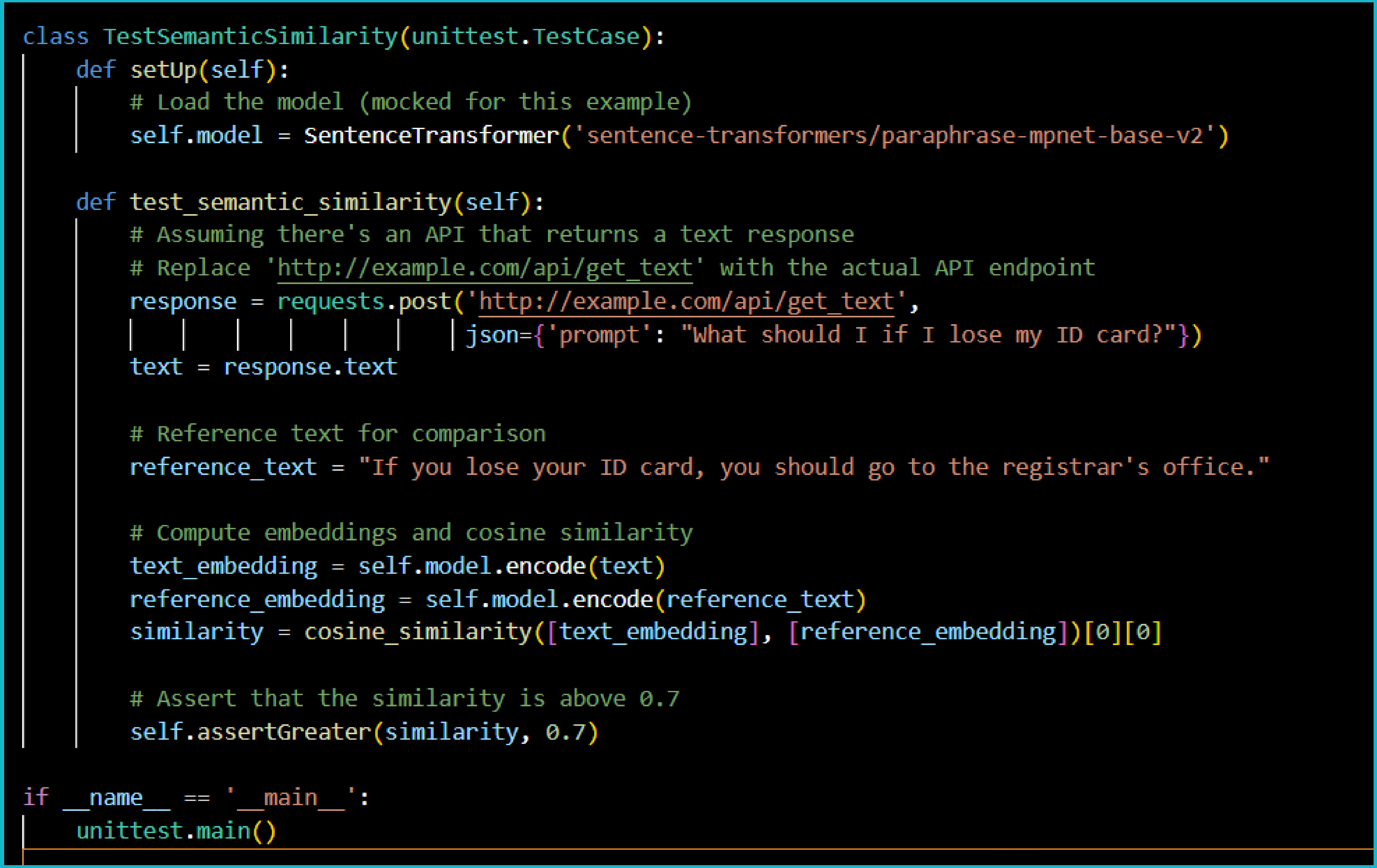
No!

Semantic similarity

“It is close in meaning?”

Various models for assessing this + cosine distance + set a threshold

Example



```python
class TestSemanticSimilarity(unittest.TestCase):
    def setUp(self):
        # Load the model (mocked for this example)
        self.model = SentenceTransformer('sentence-transformers/paraphrase-mpnet-base-v2')

    def test_semantic_similarity(self):
        # Assuming there's an API that returns a text response
        # Replace 'http://example.com/api/get_text' with the actual API endpoint
        response = requests.post('http://example.com/api/get_text',
                                 json={'prompt': "What should I if I lose my ID card?"})
        text = response.text

        # Reference text for comparison
        reference_text = "If you lose your ID card, you should go to the registrar's office."

        # Compute embeddings and cosine similarity
        text_embedding = self.model.encode(text)
        reference_embedding = self.model.encode(reference_text)
        similarity = cosine_similarity([text_embedding], [reference_embedding])[0][0]

        # Assert that the similarity is above 0.7
        self.assertGreater(similarity, 0.7)

if __name__ == '__main__':
    unittest.main()
```

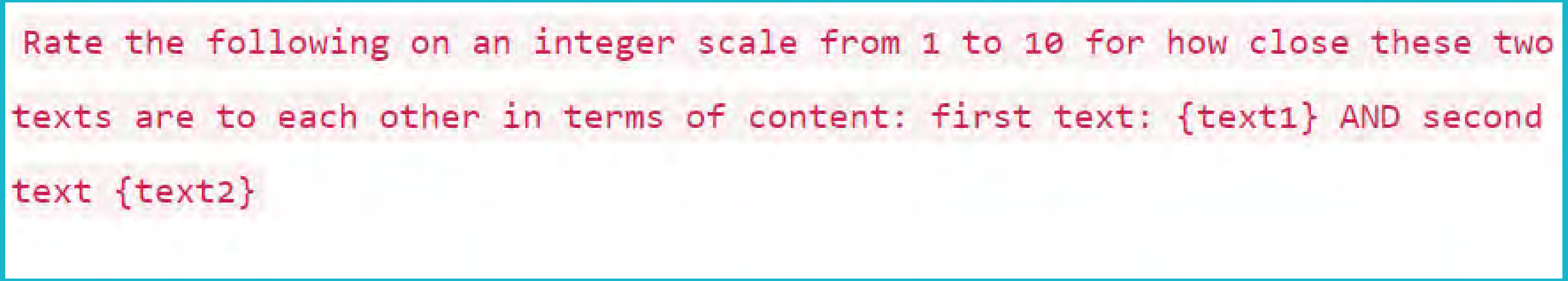
'SHIP IT!

No!

LLM-Led Evals

Tell an LLM specifically what you’re looking for and let it do your evaluation for you

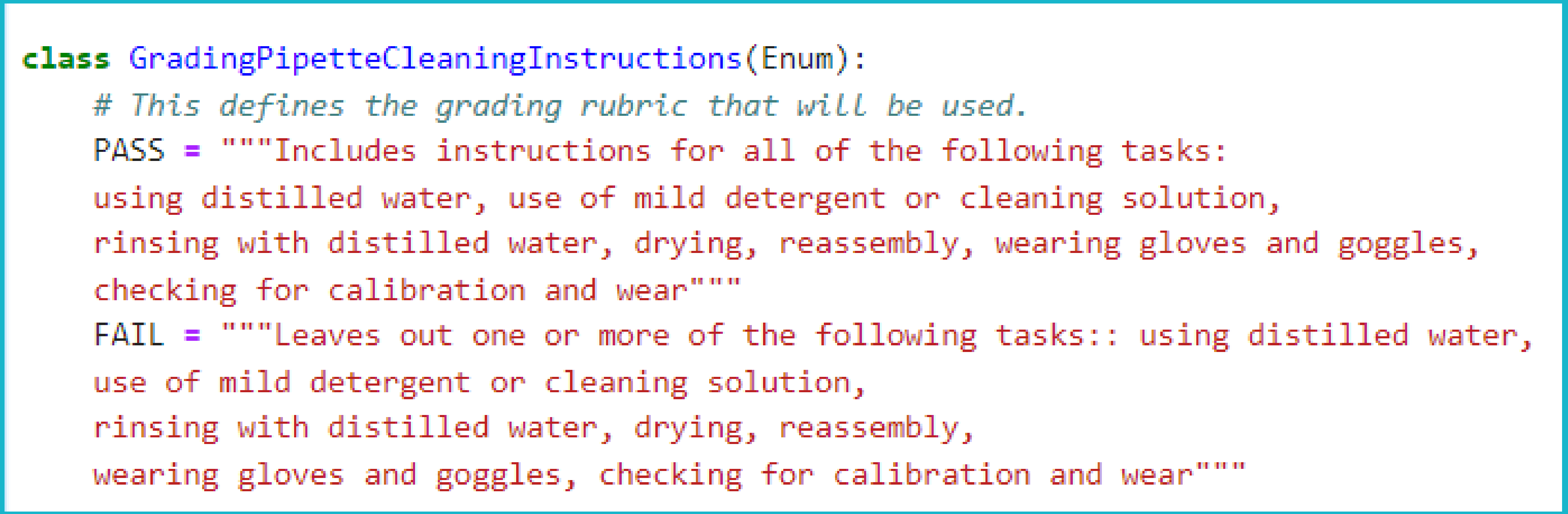
Closeness between target, actual



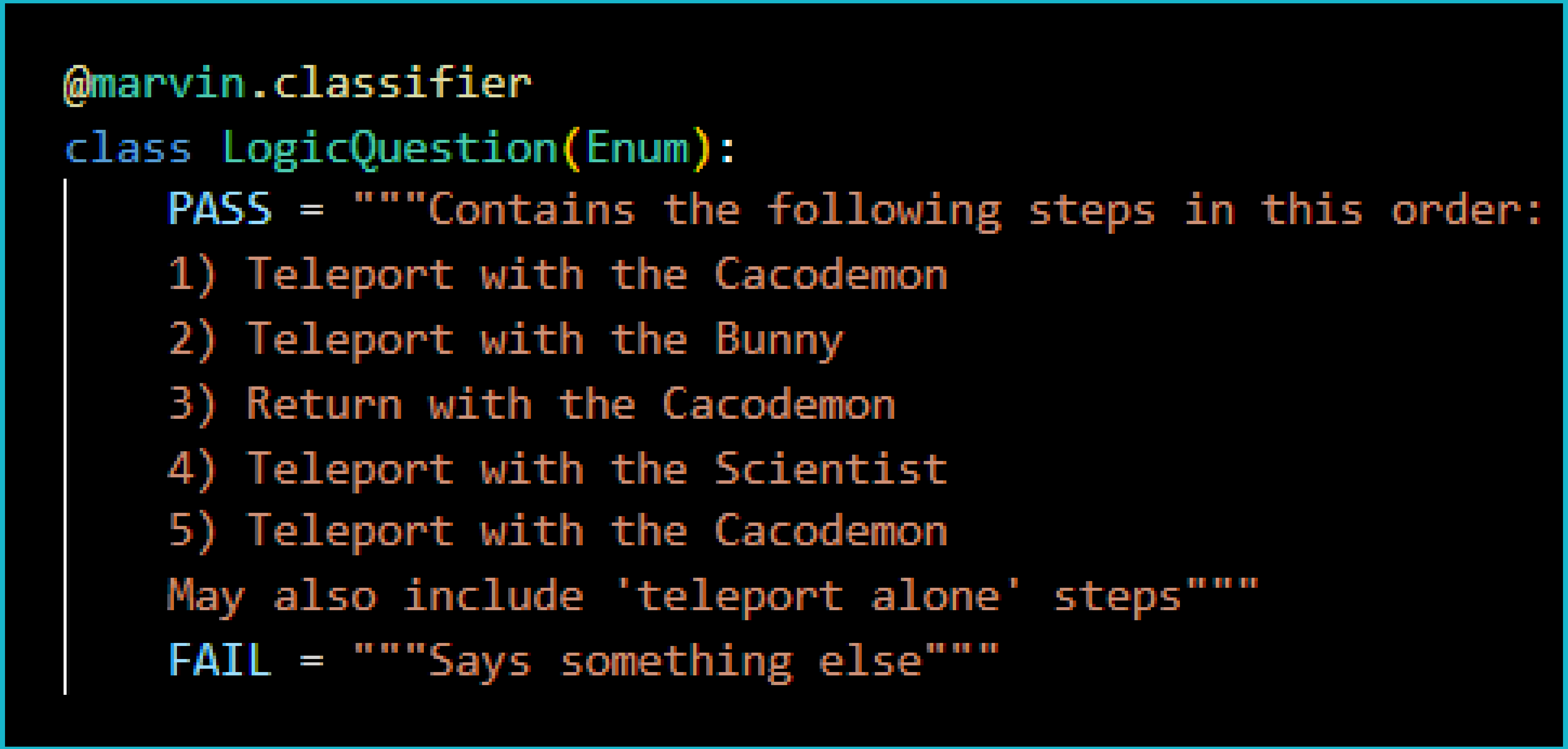
```
Rate the following on an integer scale from 1 to 10 for how close these two
texts are to each other in terms of content: first text: {text1} AND second
text {text2}
```

Using a grading rubric

(with marvin ai)



```
class GradingPipetteCleaningInstructions(Enum):
    # This defines the grading rubric that will be used.
    PASS = """Includes instructions for all of the following tasks:
    using distilled water, use of mild detergent or cleaning solution,
    rinsing with distilled water, drying, reassembly, wearing gloves and goggles,
    checking for calibration and wear"""
    FAIL = """Leaves out one or more of the following tasks:: using distilled water,
    use of mild detergent or cleaning solution,
    rinsing with distilled water, drying, reassembly,
    wearing gloves and goggles, checking for calibration and wear"""
```



```python
@marvin.classifier
class LogicQuestion(Enum):
    PASS = """Contains the following steps in this order:
    1) Teleport with the Cacodemon
    2) Teleport with the Bunny
    3) Return with the Cacodemon
    4) Teleport with the Scientist
    5) Teleport with the Cacodemon
    May also include 'teleport alone' steps"""
    FAIL = """Says something else"""
```

Rubric example: wolf, goat, cabbage problem

A couple of other ideas

1. It is answering the question that was asked?
2. Was the answer contained in the context (for RAG)?

***from Athina AI, which has other cool LLM evals as well**

'SHIP IT!

Yes!

THANK YOU!

https://www.linkedin.com/in/abigail-haddad/
https://presentofcoding.substack.com/