Module `chatbot`

Sub-modules

- chatbot.AIMLEngine
- chatbot.AzureObjectDetectionEngine
- chatbot.ClassificationEngine
- chatbot.KBEngine
- chatbot.QAEngine
- chatbot.TranslateEngine
- chatbot.WikiApi
- chatbot.YoloV5ObjectDetectionEngine

Module `chatbot.AIMLEngine`

AIML engine module is used to perform the AIML based functionalities of the chat bot. The patters of the conversation are loaded in from pre-defined in an xml file.

Functions

Function `get_response`

```
def get_response(
    query: str
) -> str
```

Get the response from the AIML agent

Args ——= **`query`** : User query

Returns: Response from AIML agent

Function `load_aiml`

```
def load_aiml(
    filepath: str
) -> None
```

Loads AIML file into the module

Args ——= **`filepath`** : Path to AIML file

Module `chatbot.AzureObjectDetectionEngine`

Functions

Function `inference_from_file`

```
def inference_from_file(
    image_path
)
```

Args ——= image_path: Returns:

Function `load_credentials`

```
def load_credentials(
    endpoint,
    subscription_key
)
```

Args ——= endpoint: subscription_key: Returns:

Module `chatbot.ClassificationEngine`

Functions

Function `classify_from_file`

```
def classify_from_file(
    img_path
)
```

Args ——= img_path: Returns:

Function `classify_from_image`

```
def classify_from_image(
    img
)
```

Args ——= img: Returns:

Function `load_model`

```
def load_model(
    model_filepath: str = '/home/ivica/Coding/uni-chat-bot/chatbot/model.h5',
    model_size: tuple = (224, 224),
    classes: list = []
)
```

Args ——= model_filepath: model_size: classes: Returns:

Module `chatbot.KBEngine`

The KBEngine module is used to provide the logical reasoning capabilities with the help of the NLTK library. Initial logic is loaded into the chatbot from the Knowledge base txt file.

Functions

Function `load_knowledge_base`

```
def load_knowledge_base(
    filepath: str
) -> None
```

Loads knowledge base from external txt file into the module

Args ——= **`filepath`** : Path to the txt KB file

Function `prove_statement`

```
def prove_statement(
    a: str,
    b: str,
    c: str
) -> bool
```

Prove statement using NLTK Inference Resolution Prover

Format for proving > a(b,c)

Args ——= **`a`** : Word

`b` Word
`c` Word

Returns ——= Validity of statement

Module `chatbot.QAEngine`

The QAEngine module is used to perform similarity-based question lookup to provide the user with the best possible answer. The similarity-based functionality is based on a set of pre-defined Q/As in a CSV file. The similarity-based component is based on the bag-of-words model, tf/idf, and cosine similarity.

Functions

Function `_get_real_question_id`

```
def _get_real_question_id(
    question: str,
    confidence_threshold: float = 0.0
) -> Tuple[bool, int]
```

Perform the similarity-based lookup for the real question from our QA list based on the user-entered question.

Similarity based lookup based on bag of words and cosine similarity is used to determine the question the user most likely wanted to ask. User question is appended to the question list and sparse matrix is created and passed to the pandas data frame. Afterwards the cosine similarity is calculated using sklearn, our question is removed from the question list and similarity list (as it's score is always 1.00). Finally, the index with biggest score is returned. Note, in order to exclude useless answers, the confidence threshold is applied.

Args —–= **`question`** : User question to apply similarity-based lookup on

`confidence_threshold` Confidence threshold for cosine-similarity. Used to exclude useless answer

Returns: Validity status, Index of question in _questions list best matching to User question input

Function `get_answer`

```
def get_answer(
    question: str,
    confidence_threshold: float = 0.25
) -> Tuple[bool, str]
```

Interface function used to obtain the answer for the question provided, running similarity-based lookup in the background.

Args —–= **`question`** : User question

`confidence_threshold` Confidence threshold for cosine-similarity. Used to exclude useless answer

Returns —–= Validity status ,answer to user question

Function `load_qa_csv`

```
def load_qa_csv(
    filepath: str
) -> None
```

Function used to load qa csv file into module

Args —–= **`filepath`** : Path to csv file

Function `load_qa_pair`

```
def load_qa_pair(
    question: str,
    answer: str
) -> None
```

Load the QA pair into QAPair module

Args ——= `question` : Question

`answer` Answer

Function `print_qa_pairs`

```
def print_qa_pairs() -> None
```

Print QA Pairs for debug purposes

Module `chatbot.TranslateEngine`

Functions

Function `load_credentials`

```
def load_credentials(
    subscription_key,
    location='global',
    endpoint='https://api.cognitive.microsofttranslator.com'
)
```

Args ——= subscription_key: location: endpoint: Returns:

Function `translate`

```
def translate(
    input_text,
    input_language='en',
    output_language='hr'
)
```

Args ——= input_text: input_language: output_language: Returns:

Module `chatbot.WikiApi`

WikiAPI Module used to interface with python wikipedia module. Used when user want to retreive the data from wikipedia on a given topic trough chat bot.

Functions

Function `get_from_wiki`

```
def get_from_wiki(
    topic: str,
    sentences=3
) -> Tuple[bool, str]
```

Get the information from wikipedia on provided topic using python wikipedia module

Args ——= `topic` : Topic of interest

`sentences` Number of sentences on the topic

Returns: Validity status, Details about the topic

Module `chatbot.YoloV5ObjectDetectionEngine`

Functions

Function `inference_from_file`

```
def inference_from_file(
    img_path
)
```

Args ——= img_path: Returns:

Function `inference_on_camera`

```
def inference_on_camera(
    camera='/dev/video0'
)
```

Args ——= camera: Returns:

Function `load_network`

```
def load_network(
    model_path,
    input_width=320,
    conf_threshold=0.25,
    iou_thres=0.45,
    classes=[]
)
```

Args ——= model_path: input_width: conf_threshold: iou_thres: classes: Returns:

Classes

Class `ObjectDetection`

```
class ObjectDetection(
    model_path,
    input_width=320,
    conf_threshold=0.25,
    iou_thres=0.45
)
```

Args ——= model_path: input_width: conf_threshold: iou_thres:

Methods

Method `detect`

```
def detect(
    self,
    main_img
)
```

Args ——= main_img: Returns:

Generated by *pdoc* 0.10.0 (https://pdoc3.github.io).