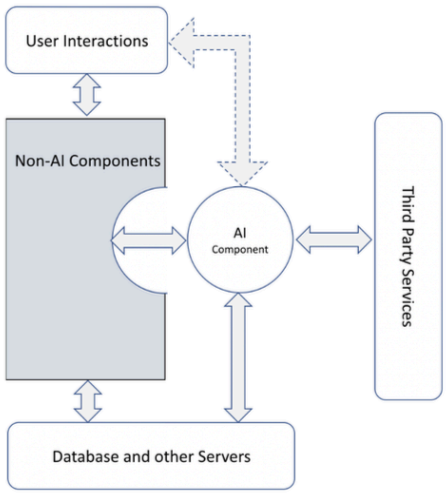
4.1 Architecture of an AI application

4.1.1 Components of an Intelligent Application and their Testing Needs

A typical AI application needs to be examined, whether it is a monolithic AI application, or an overall system comprised of a smaller set of AI components invoked from a larger non-AI application. This analysis is required to understand how various components, AI and non-AI, work together to provide the desired functionality. The interfaces between these various components need to be understood from the point of view of:

- Input data being passed
- Output generated
- Actions taken by the respective components
- Direct user interactions, if any
- Third-party system interactions, if any
- Frequency of invocation
- Number of parallel invocations, if possible
- Constraints such as
  - Timing
  - Duration
  - Ranges of inputs and outputs
- Necessary pre-conditions
- Assumptions/common settings
- Visualizing the chain of inputs and outputs with relevant transformations, if any
- Error and exceptions and their handling by
  - The component
  - Chain of components and final handling
- Logging



Intelligent App Components and Online Testing

- A typical AI Deployment Scenario
- Trained model integration phase
  - Non-AI part – usual testing

    - Integrated system – System testing
        - AI – User interactions
        - AI – Server interactions
        - Usual UI based system tests

Key desiderata for functional testing of AI systems

- The system should interpret the way humans can do in an explainable manner
    - Image data
    - Text data
    - Audio data
- Different ways of identifying test scenarios
    - Linguistic analysis
    - Q-patterns for building blocks
    - Explanations
    - Marriage of linguistics & explanations
- Metamorphic testing
    - Identifying invariances with respect to input output

Non-Functional tests

- Security
- Performance related
    - Response tie and resource usage
    - Load
    - Scalability
    - Capacity
    - Stress
- Error related
    - Fault tolerance
    - Recoverability
    - Robustness

Key desiderata in AI – Non-AI interaction

- Integration with browsers, mobile-apps and other platforms (various ways o user interaction)
    - Modes of interaction b/w AI & non-AI
        - Flag only (just feed information to the system)
        - Take action, e.g. generate response for user OR classify the problem
        - Impact on the feedback look of experience
        - Failure scenarios of interaction APIs
- Handover from AI to non-AI components and vice-versa

Furthermore, there is a need for the explainability of the reason for the system to arrive at the given solution. In addition, there is a need for identifying test scenarios for these AI systems using techniques such as linguistic analysis and exploratory testing.

combinations of AI systems. Single standalone AI systems may be insufficient to completely specify a real-world problem. To handle these scenarios, combinations of AI systems are used to model the problem. To that end, a test strategy to handle combinations of AI systems is required.

In AI - non-AI interaction, there is a need to ensure the test coverage of the AI component, the non-AI component, and the interaction involving handover between the two parts.

An example of testing such a system is provided here:
In an email application, sentence completion is provided by an AI component. The user interface of the email system represents the non-AI part and the AI component interacts with it to provide the suggested text. The non-AI part displays the suggestion and acts on the user input. If the suggestion is accepted, this might result in the storage of some data for future learning.

The same email system also provides inline spell checking, provided either by an AI or non-AI component.

If we analyze the interfaces, we find that the UI (non-AI component) is passing on the partially written text to the AI component. If there is a suggestion from the AI component, the text is displayed by the UI. Constraints related to timing and duration are that the suggestion has to be made within a few hundred milliseconds of the text being entered and this has to be a continuous process. The tester needs to determine if the system is responding fast enough. If there are timing and duration issues, such as the model predictions being slow, by the time the system shows the suggested sentence, the user may have written additional text. This may make the suggested sentence incorrect grammatically or the wrong suggestion all together.

An example of issues in the interaction of the GUI and the non-GUI (AI) parts of an output can be related to the display of the suggested text. The suggested text may be displayed at an incorrect place in the GUI. The suggested sentence needs to be visually differentiated from the actual text being written and needs to change based on user actions. Any errors in these behaviors would also fall under the GUI and non-GUI (AI) interactions.

An example of the error/exception handling part is the scenario where the sentence completion component fails (for some reason) and the GUI is able to handle it.

The settings of the email system include language settings and dictionary settings. For example, English as the language setting, and US English as the dictionary setting. The sentence completion system should provide US English suggestions. The spell and grammar check system should ideally use US English as well, in this case. Which means that the sentences formed by the AI component should not be marked incorrect text by the spell check component because of a mismatch in the assumption/settings of the two interacting AI or AI - non-AI components.

The images below show some examples of issues discovered on such a system, A and B show the marking of one name as a spelling error and not the other, whereas both names are part of the address book.

first time, the name completion happens but the name is not marked as an error for a few seconds and then it gets marked as an error. However, once marked as an error, then even deletion of the name and subsequent auto-completion marks this as an error immediately.

Image A ImageB ImageC

Analysis of interfaces

- An analysis of how various components – AI and non-AI – work together is required
- The interfaces need to be understood from the point of view
  - Input data being passed
  - Output generated
  - Actions taken by the respective components
  - Direct user interactions, if any
  - Frequency of invocation
  - Number of parallel invocations, if possible
- Necessary pre-conditions
- Visualizing the chain of inputs and outputs with relevant transformations, if any

Ensembles

- Real-world problems are too complex to be based on a monolithic AI system
  - Example: Inventory planning in retail for different items, each item category needing different models: high cost, low volume items require different items than high volume, low cost items
- Test strategy for the integration of an ensemble of AI systems (each already tested individually)
  - The first classifier identifies the category of input data
  - This classification is also a learned response
- Each AI system of the ensemble may employ any AI model/algorithm independent of its peer components

4.1.2 Interaction of AI and Non-AI Parts

In the information-oriented calls, the API or service of the AI application is invoked from an end application which can be non-AI. Typically, the AI component is invoked for a response. For example, a fraud detection algorithm works on the idea of calling a pre-defined trained model to return whether an input transaction is a fraudulent one or not. Some of the important tests for testing the interactions are related to:

- Boundary value-based tests – both input and output
- Unusual test cases (a.k.a. Corner test cases)
- Tests related to the size and type of data to be passed
- Exception handling tests
  - Response not received

- Erroneous input data tests
- Erroneous output data tests
- Request could not be completed
- Response not received
- Performance related tests
- Security related tests
- Robustness related tests

The interactions between the AI and non-AI components may have an impact on the experience of the user. The interactions can be of the following type:

- Flag only (just feed information to the system)
- Action oriented, e.g., generate a response for the user, OR classify the problem
- Scenarios of failure of interacting APIs where the APIs fail to return a result
- Handover from AI to non-AI components and vice-versa

When tests are designed for the AI systems, these are of following test levels:

- Tests that just test the AI part
- Tests that just test the non-AI part
- Tests for integration of both parts
- Cached response vs. learned responses

In a larger non-standalone system, there is a need to look at the coverage of the deployed AI model as well as the performance management of the deployed AI model. After the development of AI component, the system needs to be tested in a deployment environment.

4.2 Linguistic Analysis Test Design Method

4.2.1 Linguistic Analysis-Based Test Design

Linguistic Analysis is used to design a large number of scenarios even when the requirements are poorly documented [LA1]. A linguistic analysis of the requirements identifies the test objects, and the actions to be taken on them. This method helps finding corner (unusual) test cases, a key requirement for testing AI systems.

The method works as below:

1. Identify the nouns which represent the test objects and the verbs that represent the actions.
2. Identify the properties of the noun.
3. Identify the properties of the properties and repeat it until no more properties can be discovered or you think sufficient depth has been reached.
4. Identify adverbs and adjectives applicable to the nouns and verbs to identify more properties.
5. Use 5W1H (What, Why, Where, When, Which, How) on the verbs to identify the noun-verb combinations giving you the functional and non- functional tests.

a. Who/what is the agent?
b. What are the instruments/means to accomplish the scenario?
c. What is the purpose of the action?
d. What is the direction of the action?
e. What is the source of the action?
f. What is the motive of the action?
g. Who possesses the means and the results of the action?
h. Where does the action take place?
i. Who is the receiver of the action?

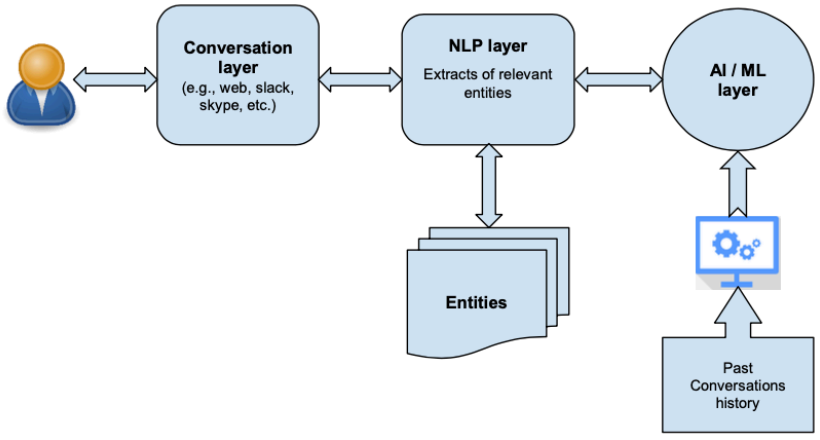
These questions follow the grammar rules of Sanskrit [LA2].
In case of testing AI systems, the data is the test case. The given method allows the identification of both, the data as well as the combination of actions (scenarios).

4.3 Testing AI systems

4.3.1 Test a Chatbot

Chatbots are used today in several contexts, ranging from customer service bots, information bots, to support bots.

The logical architecture shows the key layers of a chatbot, including the conversation layer, NLP layer, AI/ML layer and connectivity layer. The conversation layer manages the front-end interaction with the end user for example, Skype, Slack, Facebook messenger, or a web frontend. The NLP layer extracts the relevant entities and information from the end user utterances. The AI/ML layer decodes the actual intent of the utterance, having been trained on past historical conversations fed as data to the system.



Test cases should cover input data variations at each individual layer.
The conversation layer needs to be tested for the right data transmission from front-end to the backend and vice-versa. At NLP layer, one should test if the data-cleaning and preprocessing of raw input text utterances is happening properly and if the extracted entities are enough to convey the context in an ongoing conversation. The AI/ML layer or model (trained on historical conversations) should be tested for its accuracy of predicting

/ 'Hello' / 'Greetings', etc. 'Welcome' intent should be predicted.

In addition, the chatbot should be able to jump between various intents seamlessly if the conversation flow sequence changes abruptly. So, the test cases covering dynamic switching of intent should be identified.

Nonfunctional requirements are to be tested for the performance of the frontend primarily, whether it is able to withstand many concurrent customer invocations.