Development of a Cognitive Assistant to Learn Concepts for Placement Assistance

Dhana Lakshmi R[1], Abirami S[2], and Srivani M[3]
{dhanalakshmird03@gmail.com[1]}

CEG Anna University[1], CEG Anna University[2], CEG Anna University[3]

Abstract. A cognitive assistant helps the humans and enhance their capabilities to solve a large range of complex tasks. The main aim of this paper is to develop a pedagogical cognitive assistant to improve the reasoning abilities and decision making skills. The proposed system has been implemented to assist as a personal agent for students to learn python programming language. The cognitive assistant facilitates natural interactions with the students and it applies human reasoning skills to judge the students ability and train them further. The proposed techniques include Question Answer (QA) analyser, dynamic study plan generation by using assertion graph and accurate answer generation by using evidence extraction and inference generation. A cognitive conversation increases user's satisfaction and easily engages them with the system and it has achieved significant higher learning gains than a non-interactive online course. The proposed system is evaluated by using the confidence weighted score evaluation metric.

Keywords: Cognitive Assistants, human reasoning skills, question answering systems, evidence extraction

1. Introduction

Cognitive computing refers to an intelligent system that learns at scale, reason with purpose and interact with humans and other smart systems naturally. An important feature of cognitive systems is to learn from arriving data and from their communications with humans. The collaboration of cognitive systems and humans opens new potential to produce better products taking into account the combination of logical ability and encyclopedic information of computers, creativity, moral and expertise of humans.

A conversational agent is a system that perform best attempt to maintain a discussion, with personal assistants that understand user's requests and perform tasks on their behalf. Understanding natural language involves understanding grammars and semantics to recognize important words from the input [7]. The challenges are to reduce the maintenance cost of the system, handle huge knowledge bases, training the cognitive system. For example, the systems have to be tailored to understand new domain concepts and integrate them for reasoning. Another challenge is introducing social dialogue into the system[8][9].

Social talk, also called social dialogue, is commonly perceived as little complicated chat whose principle is not to exchange the content but provide natural interactions [6]. Social conversations have an important role in the area of development of conversational systems to launch the social interaction with users in order to help human and computer collaboration and maintain their confidence level.

Existing placement assistance principle is to exchange the concept, but no system-user interactions. Placement assistance has prefixed set of concepts, which are used to answer the questions. The main contribution of the system consists in proposing a modular architecture, and allows adapting conversational agents in a modular way, by combining question answering system, social and proactive dialogue capabilities independently, i.e. without make changes on any other modules. Advantages are user friendly conversation, easy maintainability and domain concepts portability.

2. Background and Related Work

2.1 Improving Human Reasoning Skills

Nguyen-Thinh Le and Laura Wartschinski [3] developed a pedagogical agent which aims to improve the reasoning abilities and decision making capability of the users. This cognitive assistant is able to hold conversation with users in natural language in order to help them solve problems of common heuristics and biases. LIZA cognitive assistant could test persons, improve their reasoning skills, and show significant higher learning gains than a non-interactive online course. The natural language conversation helps LIZA to train human reasoning capability using Bayesian reasoning. The advantage of the system is that it supports humans and enhances their capabilities to

The limitation is that long-standing effects could be examined by reassessing months or even years.

2.2 Learning Java Featuring Social Dialogue

Miguel Coronado et al. [1] implemented a modular cognitive agent for question answering and social dialogues improvement for a particular domain. Conversational agent has been developed for maintaining a conversation process, with the personal assistant that understand the user's requests and execute the respective process on their behalf. The advantage of the system is to increase user's satisfaction and makes them easily engage with the system. The limitation is that question answering systems and a personal assistant does not generate a social dialogue, instead it boosts user's engagement and increases user's satisfaction. This increases the maintenance cost of natural language systems, facilitates more flexible dialogue and enhances the document library by covering as many questions as possible.

2.3 Watson Paths

Adam Lally et al. [2] proposed a question answering system based on Watson. The proposed system takes a number of questions, which are natural language oriented as input and particular answers are generated along with perfect confidences score as output. The system uses the scenario-based questions and the answer with proper evidence is contained in evidence source. Instead, for many scenario based questions, information from multiple documents and other sources must be retrieved and then integrated to answer the questions properly. The main challenges for a descriptive question answer system are retrieving the correct document, and then extracting the correct answer from that document. Question answer system not only returns the correct answer but should also give correct explanation to the user.

3 System Architecture

The proposed system acts as a cognitive assistant to improve the domain knowledge of urban area student. Figure 1 depicts the system architecture of cognitive assistant. It consists of three modules namely QA analyzer, knowledge base and cognitive reasoning. The primary motivation of the system is to develop the student domain knowledge for placement assistance.

The input of the system is the user response (answer) to the question which is asked by the QA analyzer. The system's first step is to calculate the user knowledge about the domain. User is then exposed to a number of questions. Based on user response (answer) word net similarity is calculated using Wu & Palmer algorithm. The knowledge base is constructed for feeding the domain knowledge to both QA analyzer and assertion graph. The knowledge base for QA analyzer has a list of task questions, each linked with particular topic, suitable concept (correct answer) and suitable justification.

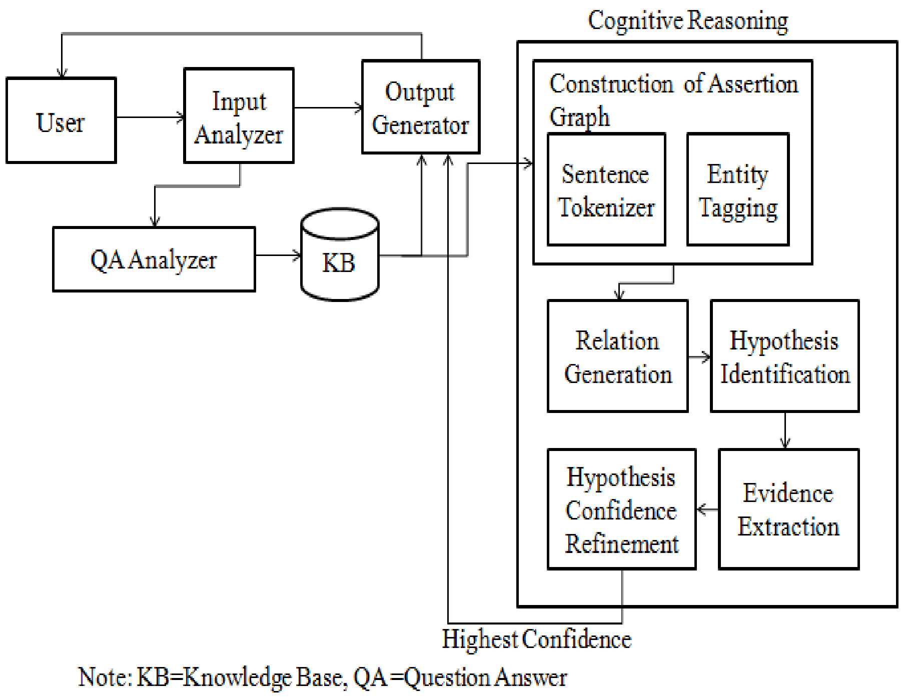


Fig.1. System Architecture for Cognitive Assistant

The cognitive reasoning creates a dynamic study plan to the user based on their level of understanding. The cognitive reasoning step deals with the construction of assertion graph, hypothesis identification, evidence extraction and inference generation. The assertion graph is used to represent the concepts in the domain of python, where the users have to gain more knowledge in. The assertion graph consists of sub-modules, namely sentence tokenization, entity tagging and relation generation.

The hypothesis identification module is defined as the list of user's answer (hypothesis). Evidence extraction module is used to extract the evidence for teaching the user's wrong answer with the proper evidence. Evidence is extracted from the concept network. The concept network is represented by a graph format which has important concepts and their semantic relationship. Finally the inference is generated. Inference is the right answer with right evidence. Inferences are generated by splitting the punch line question, extracting the proper evidence from the concept network and forming final inference to the wrong answer's punch line question.

4 Methodology

The proposed framework consists of the following modules

- QA Analyzer
- Construction of Assertion Graph
- Evidence Extraction and Inference System

4.1 QA Analyzer

QA analyzer module consists of knowledge analysis, answer evaluation using Wordnet-wup similarity and answer validation using similarity ratio. The system begins its conversation with the user through some greeting and asks the task questions for analyzing the user's domain knowledge. Task questions are the basics questions from the various concepts in that domain. These task questions are mapped with their topic, which is one of the concepts in that domain and with their answer, which is the description of one of the concept in that domain and with their justification. This justification gives the suggestion to the task questions which is wrongly answered by the users. These are stored in the knowledge base for QA analyzer. The answer evaluation is the main step in QA analyzer, where the user's answers and the answers for those particular questions are evaluated in terms of semantic similarity.

ALGORITHM 1

```
INPUT: User's Answers to Questions.
OUTPUT: Estimation of User's Knowledge Level.
1: for each User's Answers do
2:     Find the synsets for each Users Answers
3:        Compare the two synsets with wup_similarity and store similarity ratio as a result for two synsets.
4:        if similarity ratio1 < similarity ratio2 then
5:        Return 1(similarity ratio2 and its text)
6:        else
7:        Return 0(similarity ratio1 and its text)
8:        end if
9: end for
```

The algorithm used in the answer evaluation is the Wu & Palmer – Words Similarity algorithm. It calculates the similarities based on the similarity of the senses of the word and their synsets. The answer validation deals with checking whether the user's answer and actual answer to the question is similar in the basics of the semantic similarity between them. If the validation metric has not met the threshold value, then it does not move to next topic and the user's capability label is generated. This indicates that the user does not have enough knowledge. The following Algorithm 1 describes the steps involved in QA analyzer.

4.2 Construction of Assertion Graph

Construction of Assertion Graph consists of sentence extraction, entity tagging and learner reassessment. Initially, the domain concepts or python concepts are derived from the python books. The sentence tokenization algorithm is used to split the large paragraphs in books into a number of statements. To create the nodes for assertion graph, the important concepts are extracted from the splitted statements. The entities are extracted by entity tagging algorithm. The node prioritization step is used to prioritize the entities and generate the nodes for the graph.

ALGORITHM 2

```
INPUT: Large Passage Contains Concepts of Python String.
OUTPUT: Assertion Graph.
1: Calculate sent_tokenize for passage and Return splitted text.
2: for each splitted text do
3:         Extract subject, root and object from each splitted text.
4:         Declare subject, root and object as nodes in graph.
5:         Create graph G in Networkx.
6:      Each node is added in graph by add_edge.
7:    Compose the each graph into a graph by nx.compose.
8: end for
9: Draw graph by nx.draw_networks
```

Then, the assertion graph is generated by using spacy dependency graph algorithm. This algorithm uses the nodes and their relationship for creation of assertion graph. The dynamic study plan is generated by using assertion graph with user's capability label. Questions are asked to verify whether the user have studied the dynamic study plan or not. The knowledge base is constructed for assertion graph by using ontology. Knowledge base consists of lists of concepts (nodes). Each concept is mapped with question and answer to that question. The following Algorithm 2 describes the steps involved in construction of assertion graph.

4.3 Evidence Extraction and Inference System

Evidence Extraction and Inference System consists of list of punch line questions, hypothesis identification, evidence extraction and inference system. The lists of punch line questions are asked to the user continuously without evaluation of their answers. The user's answers to the list of questions are made as the hypothesis collections or list of hypothesis. Then the user's answers are evaluated using the Wu & Palmer algorithm.

The validation is done between the list of hypothesis and the actual answers for questions. If the process is valid, then the answer is correct and the user acquires the domain knowledge. Otherwise, the answer is not correct and the user does not acquire the domain knowledge. The inference system considers the particular question and the evidence is extracted from the concept network.

ALGORITHM 3

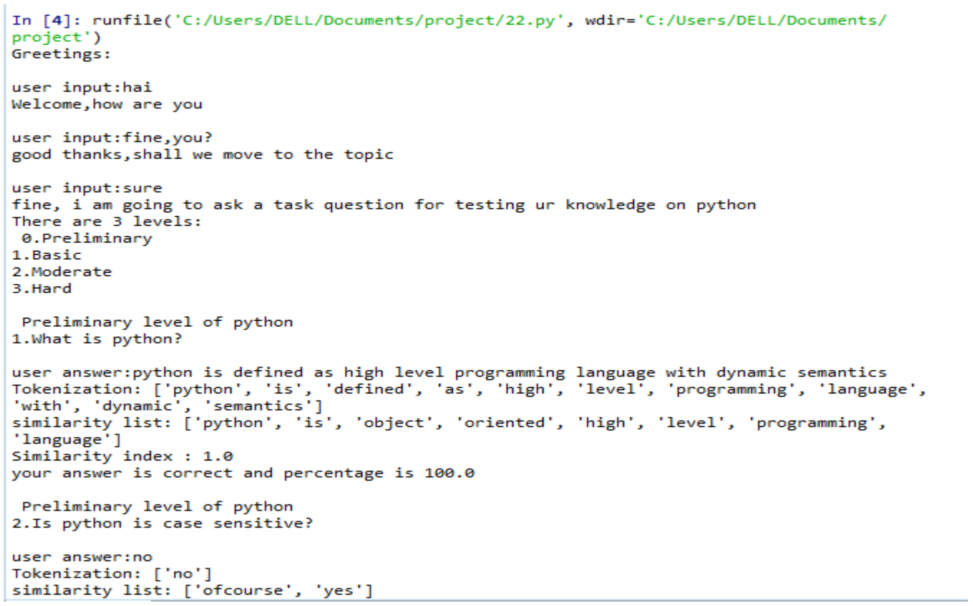
```
INPUT: List of Hypothesis.
OUTPUT: Inference to Each Punch Line Questions.
1: Assign the list of user's answers to list of hypothesis.
2: for each hypothesis do
3: Find the synsets for each hypothesis
4:         Compare the each two synsets with wup_similarity and store similarity ratio as result for two
           synsets.
5:         Identify the wrongly answered questions.
6:         Split the wrongly answered questions by strip ().
7:      Extract evidence for each spitted questions.
8:      Calculate inference from evidence for those questions.
9:    end for
10: Pictorials representation of inference by Networkx graph.
```

The particular questions are splitted into a number of sub-questions and for those sub-questions evidence is extracted from the concept network. Finally, all the inference is combined to form a final inference for that particular question. This procedure is repeated for all questions whose answers are wrongly answered by the user. The following Algorithm 3 describes the steps involved in Evidence Extraction and Inference System. This algorithm's input is list of hypothesis and output is inference to each punch line questions.

5 Discussion and Results

The implementation and the result of the cognitive system's input is user's answer as the response and output is the understanding of user's capability level. The proposed framework provides the respective dynamic study plan and also provides the inference to the question which is not understood by the user during the learning phase.

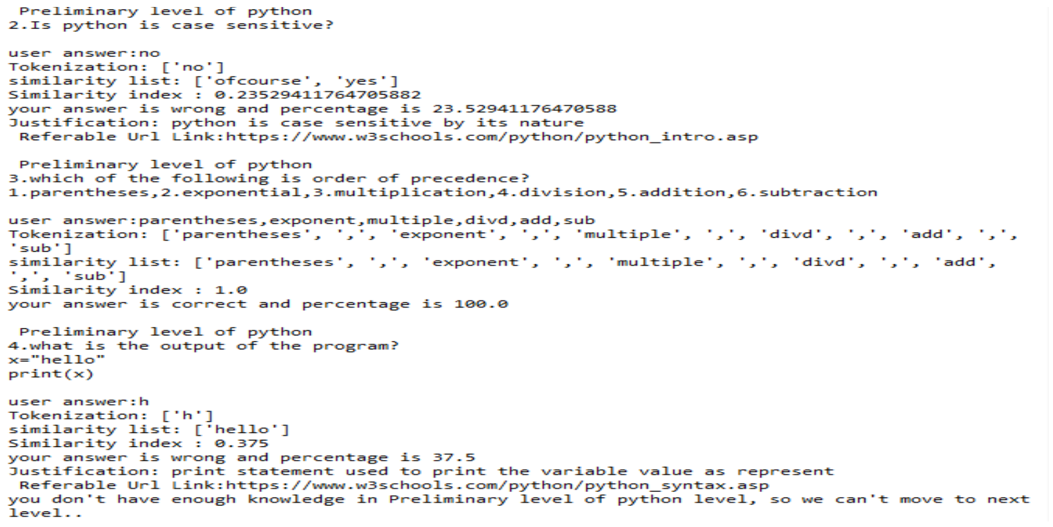
The QA analyzer Figure 2 consists of greeting, task questions, user's answer for that task questions, similarity between the user answer and actual answer.



```
In [4]: runfile('C:/Users/DELL/Documents/project/22.py', wdir='C:/Users/DELL/Documents/
project')
Greetings:

user input:hai
Welcome,how are you

user input:fine,you?
good thanks,shall we move to the topic

user input:sure
fine, i am going to ask a task question for testing ur knowledge on python
There are 3 levels:
 0.Preliminary
1.Basic
2.Moderate
3.Hard

 Preliminary level of python
1.What is python?

user answer:python is defined as high level programming language with dynamic semantics
Tokenization: ['python', 'is', 'defined', 'as', 'high', 'level', 'programming', 'language',
'with', 'dynamic', 'semantics']
similarity list: ['python', 'is', 'object', 'oriented', 'high', 'level', 'programming',
'language']
Similarity index : 1.0
your answer is correct and percentage is 100.0

 Preliminary level of python
2.Is python is case sensitive?

user answer:no
Tokenization: ['no']
similarity list: ['ofcourse', 'yes']
```

Fig.2. QA Analyzer-Greetings

The QA analyzer Figure 3 consists of task questions, their answer provided by the user, similarity ratio between user's answer and actual answer. If the similarity ratio is higher than the threshold value then the user moves to next topic, otherwise inference is generated and rendered to the user.



```
 Preliminary level of python
2.Is python is case sensitive?

user answer:no
Tokenization: ['no']
similarity list: ['ofcourse', 'yes']
Similarity index : 0.23529411764705882
your answer is wrong and percentage is 23.52941176470588
Justification: python is case sensitive by its nature
 Referable Url Link:https://www.w3schools.com/python/python_intro.asp

 Preliminary level of python
3.which of the following is order of precedence?
1.parentheses,2.exponential,3.multiplication,4.division,5.addition,6.subtraction

user answer:parentheses,exponent,multiple,divd,add,sub
Tokenization: ['parentheses', ',', 'exponent', ',', 'multiple', ',', 'divd', ',', 'add', ',',
'sub']
similarity list: ['parentheses', ',', 'exponent', ',', 'multiple', ',', 'divd', ',', 'add',
',', 'sub']
Similarity index : 1.0
your answer is correct and percentage is 100.0

 Preliminary level of python
4.what is the output of the program?
x="hello"
print(x)

user answer:h
Tokenization: ['h']
similarity list: ['hello']
Similarity index : 0.375
your answer is wrong and percentage is 37.5
Justification: print statement used to print the variable value as represent
 Referable Url Link:https://www.w3schools.com/python/python_syntax.asp
you don't have enough knowledge in Preliminary level of python level, so we can't move to next
level..
```

Fig.3. QA Analyzer

Figure 4 shows the assertion graph for python concepts. The large paragraphs of python concepts are parsed through sentence tokenization, entity tagging and node prioritization. The outcomes are entity or nodes for the graph construction. Figure 5 shows the assertion graph for python string concepts.

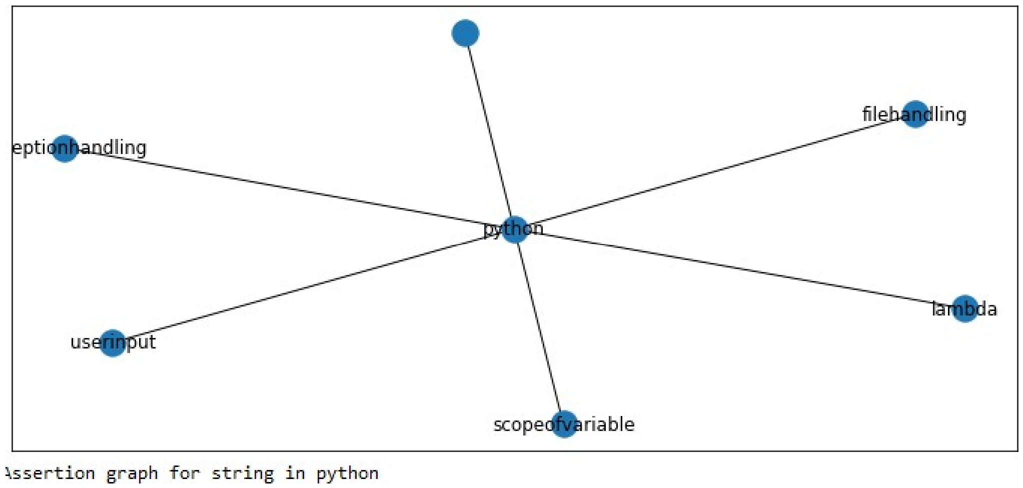
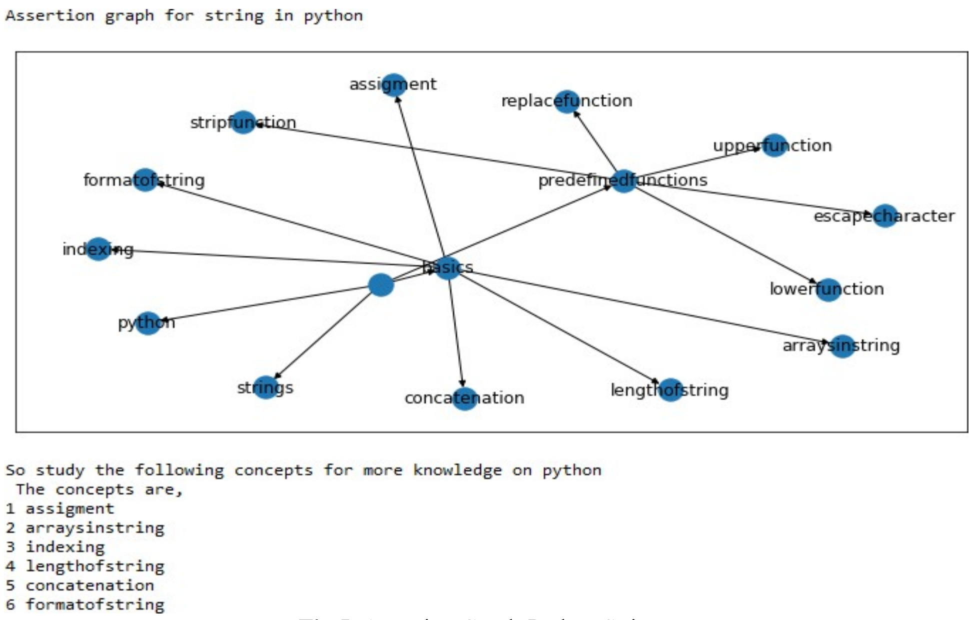


Fig.4. Assertion Graph-Python

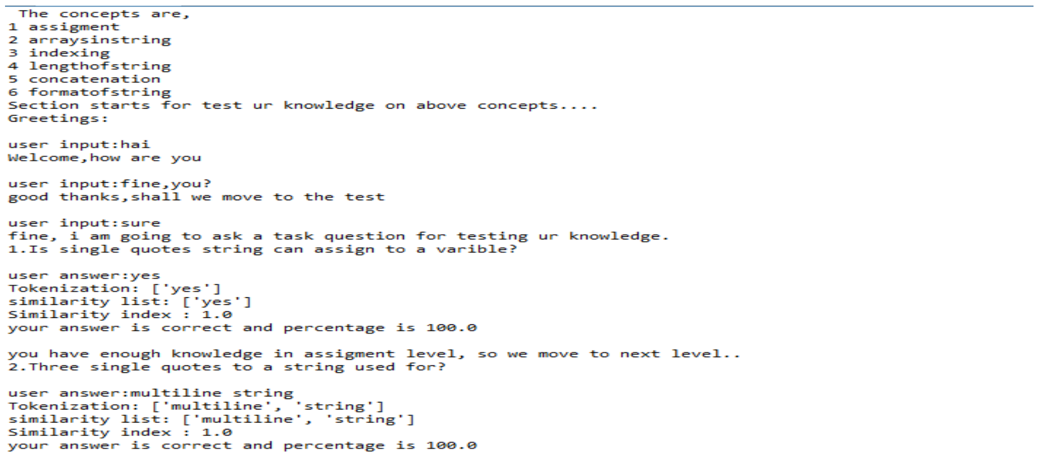
The system provides the list of concepts or topics in python string which are the topic that are not correctly answered by the user during the QA analyzer module. The user has to study the list of topics by their own and the system will check or verify whether the user has studied the list of topics or not by using learner's reassessment.



```
So study the following concepts for more knowledge on python
 The concepts are,
1 assigment
2 arraysinstring
3 indexing
4 lengthofstring
5 concatenation
6 formatofstring
```

Fig.5. Assertion Graph-Python String

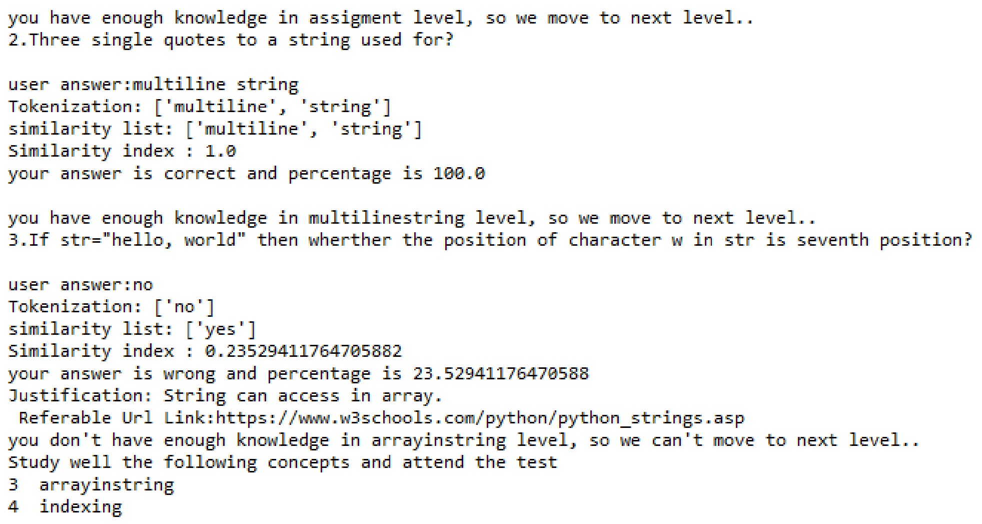
Dynamic plan creation Figure 6 and Figure 7 shows the list of topics that should be read by the user. Some topic questions are asked by the system to the user to verify their topic knowledge after completion of their self-study.



```
 The concepts are,
1 assigment
2 arraysinstring
3 indexing
4 lengthofstring
5 concatenation
6 formatofstring
Section starts for test ur knowledge on above concepts....
Greetings:

user input:hai
Welcome,how are you

user input:fine,you?
good thanks,shall we move to the test

user input:sure
fine, i am going to ask a task question for testing ur knowledge.
1.Is single quotes string can assign to a varible?

user answer:yes
Tokenization: ['yes']
similarity list: ['yes']
Similarity index : 1.0
your answer is correct and percentage is 100.0

you have enough knowledge in assigment level, so we move to next level..
2.Three single quotes to a string used for?

user answer:multiline string
Tokenization: ['multiline', 'string']
similarity list: ['multiline', 'string']
Similarity index : 1.0
your answer is correct and percentage is 100.0
```

Fig.6. List of Concepts

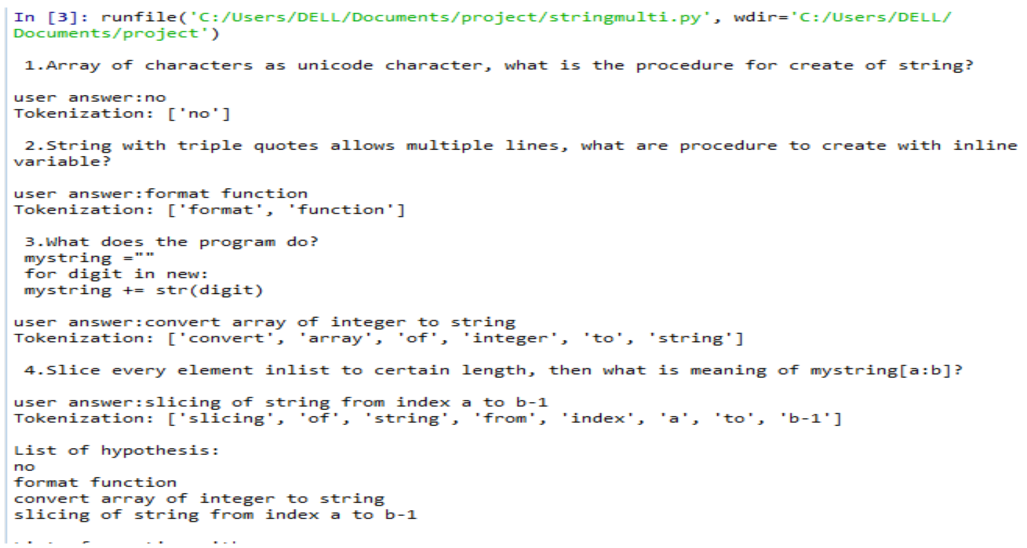
The dynamic plan creation begins with the topic questions that are made under a specific topic. The topic questions are asked to the user in one by one manner. Each question is asked and if the answer is correct then the next topic question will be provided. Otherwise, the system shows dynamic study plan to each user with their understanding level during the self-study phase.



```
you have enough knowledge in assigment level, so we move to next level..
2.Three single quotes to a string used for?

user answer:multiline string
Tokenization: ['multiline', 'string']
similarity list: ['multiline', 'string']
Similarity index : 1.0
your answer is correct and percentage is 100.0

you have enough knowledge in multilinestring level, so we move to next level..
3.If str="hello, world" then wherther the position of character w in str is seventh position?

user answer:no
Tokenization: ['no']
similarity list: ['yes']
Similarity index : 0.23529411764705882
your answer is wrong and percentage is 23.52941176470588
Justification: String can access in array.
 Referable Url Link:https://www.w3schools.com/python/python_strings.asp
you don't have enough knowledge in arrayinstring level, so we can't move to next level..
Study well the following concepts and attend the test
3  arrayinstring
4  indexing
```

Fig.7. Dynamic Plan Creation

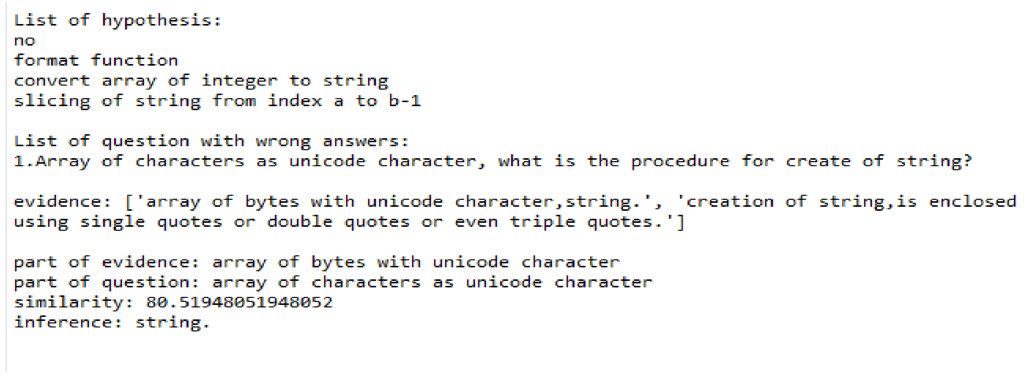
The dynamic plan creation generates the dynamic self-study plan for each user, after that self-study phase gets over. Figure 8 describes the hypothesis identification that shows the list of punch line question that has to be asked by the system to the user. But in a different way, all punch line questions are asked line by line without any evaluation of the user's answer. After the user has answered all the questions, the list of answers is rendered as a list of hypothesis.



```
In [3]: runfile('C:/Users/DELL/Documents/project/stringmulti.py', wdir='C:/Users/DELL/
Documents/project')

 1.Array of characters as unicode character, what is the procedure for create of string?

user answer:no
Tokenization: ['no']

 2.String with triple quotes allows multiple lines, what are procedure to create with inline
variable?

user answer:format function
Tokenization: ['format', 'function']

 3.What does the program do?
 mystring =""
 for digit in new:
 mystring += str(digit)

user answer:convert array of integer to string
Tokenization: ['convert', 'array', 'of', 'integer', 'to', 'string']

 4.Slice every element inlist to certain length, then what is meaning of mystring[a:b]?

user answer:slicing of string from index a to b-1
Tokenization: ['slicing', 'of', 'string', 'from', 'index', 'a', 'to', 'b-1']

List of hypothesis:
no
format function
convert array of integer to string
slicing of string from index a to b-1
```

Fig.8. Hypothesis Identification

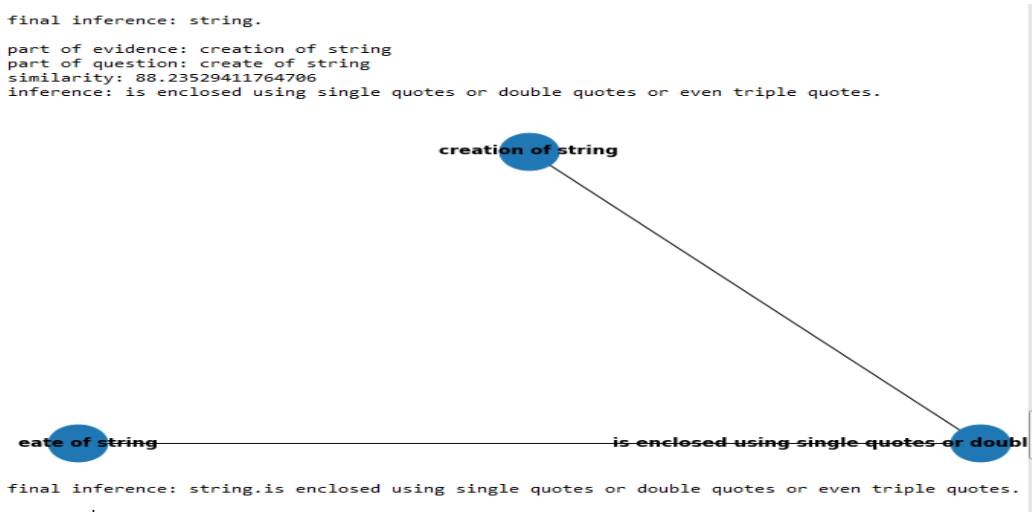
After the list of hypothesis is generated, then the answer is validated. If the user's answer is not correct then the system will provide the answer with inference to that particular question.



```
List of hypothesis:
no
format function
convert array of integer to string
slicing of string from index a to b-1

List of question with wrong answers:
1.Array of characters as unicode character, what is the procedure for create of string?

evidence: ['array of bytes with unicode character,string.', 'creation of string,is enclosed
using single quotes or double quotes or even triple quotes.']

part of evidence: array of bytes with unicode character
part of question: array of characters as unicode character
similarity: 80.51948051948052
inference: string.
```

Fig.9. List of Wrongly Answered Questions

The hypothesis identification module provides the output as a list of hypothesis and list of questions which are wrongly answered by the user. The Figure 9 shows the evidence which is extracted from the concept network. Figure 10 show the inference to the question in a graphical representation. To improve the result of the system, the performance evaluation is done by the confidence weighted score.



```
final inference: string.

part of evidence: creation of string
part of question: create of string
similarity: 88.23529411764706
inference: is enclosed using single quotes or double quotes or even triple quotes.
```

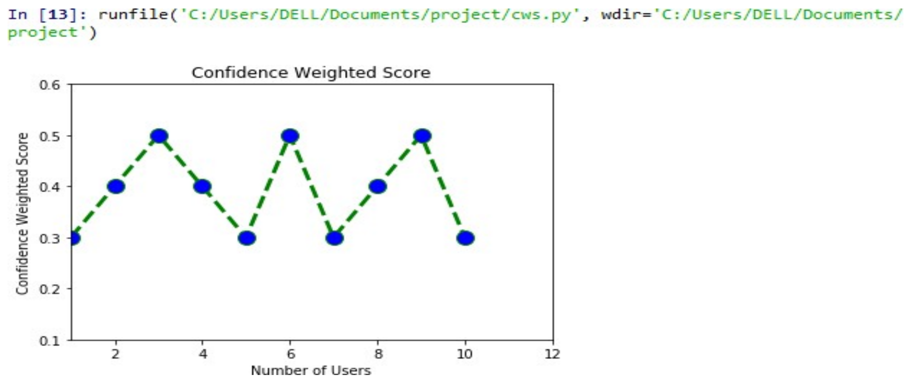
```
final inference: string.is enclosed using single quotes or double quotes or even triple quotes.
```

Fig.11. Final Inference System

Equation 1 describes the confidence weighted score. It is a metric that evaluates the accuracy of the system and its confidence on producing the top answer to that particular question. Sort the entire question and their relevant top answer pairs in descending order. The answer with the highest confidence score is the correct answer. Formula is

$$\mathrm{CWS} = \frac{1}{\mathrm{n}} \sum_{\mathrm{i}=0}^{\mathrm{n}} \frac{\text{number correct in first i rank}}{\mathrm{i}} \quad (1)$$

Where CWS is confidence weighted score, n is the number of questions asked to the user and i indicates each question in rank.



```
In [13]: runfile('C:/Users/DELL/Documents/project/cws.py', wdir='C:/Users/DELL/Documents/
project')
```

Fig.12. Confidence Weighted Score

Figure 12 shows the graphical representation of confidence weighted score where x-axis indicates the number of users and y-axis indicates the confidence weighted score. For each user, the confidence is calculated by considering their answer or by considering the list of hypothesis their similarity ratio is calculated. Rank the list of hypothesis with the help of their similarity ratio.

6 Conclusion

The proposed framework deals with the development of a cognitive assistant to learn concepts for placement assistance using dynamic study plan, and evidence based inference system. This will enable the system to understand the user's domain knowledge by asking a number of questions in the domain, where the user has to develop their knowledge on. The limitations of this system are that the questions are only factoid questions and the hypothesis set requires more possible answers to the question. The knowledge level of user on the domain can be improved by posing programmatic questions. Future works are to upgrade the hypothesis generation by context detection using machine learning techniques. For evidence extraction, many more evaluation based confidence metrics can be considered for improving the accuracy of the system.

References

1. **Journal article**: Coronado M, Iglesias C A, Carrera A, & Mardomingo A, “A cognitive assistant for learning java featuring social dialogue”, in the proceedings of Human-Computer Studies, Vol. no. 117, pp. no. 55-67 (2018).
2. **Journal article**: Lally A, Bagchi S, Barborak M A, Buchanan D W, Chu-Carroll J, Ferrucci D A & Patwardhan S, “WatsonPaths: scenario-based question answering and inference over unstructured information”, in the proceedings of AI Magazine, Vol. no. 38, pp. no. 59-76 (2017).
3. **Journal article**: Le, N T, & Wartschinski L, “A Cognitive Assistant for improving human reasoning skills”, in the journal of Human-Computer Studies, Vol. no. 117, pp. no. 45-54 (2018).
4. **Journal article**: Lovenia H, Limanta F, & Gunawan A, “Automatic Question-Answer Pairs Generation from Text” (2017).
5. **Journal article**: Santos J, Rodrigues J J, Casal J, Saleem K & Denisov V, “Intelligent personal assistants based on internet of things approaches”, in the proceedings of IEEE Systems Journal, Vol. no. 12, pp. no. 1793-1802 (2016).
6. **Journal article**: Tibor Kocsis, Stephane Negny, Pascal Floquet, Xuan Meyer, Endre Rev, “Case-Based Reasoning system for mathematical modelling options and resolution methods for production scheduling problems: Case representation, acquisition and retrieval”, in the proceedings of computers and industrial engineering, Vol. no. 77, pp. no. 46-64 (2016).
7. **Journal article**: Weidong Huang, Xiang Xiao, Mengqiang Xu, “Design and implementation of domain-specific cognitive system based on question similarity algorithm”, in the proceedings of Cognitive system research, Vol. no. 57, pp. no. 20-24 (2018).
8. **Journal article**: Weiguo Zheng, Hong Cheng, Jeffrey Xu Yu, Lei Zou, Kangfei Zhao, “Interactive natural language question answering over knowledge graphs”, in the proceedings of information sciences, Vol. no. 481, pp. no. 141-159 (2019).
9. R. Joseph Manoj, M. D. Anto Praveena, K. Vijayakumar, “An ACO–ANN based feature selection algorithm for big data”, Cluster computing, springer,march 2018.
10. Dafni Rose J, Vijayakumar K, “Data Transmission Using Multiple Medium Concurrently”, IJET,2018.