IoTutor: How Cognitive Computing Can Be Applied to Internet of Things Education

Suejb Memeti, Sabri Pllana, Mexhid Ferati, Arianit Kurti, Ilir Jusufi

IoTutor: How Cognitive Computing Can Be Applied to Internet of Things Education *

Suejb Memeti[1], Sabri Pllana[1], Mexhid Ferati[2], Arianit Kurti[1,3], and Ilir Jusufi[1]

[1] Linnaeus University, Department of Computer Science and Media Tech., Sweden
{suejb.memeti, sabri.pllana, arianit.kurti, ilir.jusufi}@lnu.se
[2] Linnaeus University, Department of Informatics, Sweden
mexhid.ferati@lnu.se
[3] RISE Research Institutes of Sweden, Sweden
arianit.kurti@ri.se

Abstract. We present IoTutor that is a cognitive computing solution for education of students in the IoT domain. We implement the IoTutor as a platform-independent web-based application that is able to interact with users via text or speech using natural language. We train the IoTutor with selected scientific publications relevant to the IoT education. To investigate users' experience with the IoTutor, we ask a group of students taking an IoT master level course at the Linnaeus University to use the IoTutor for a period of two weeks. We ask students to express their opinions with respect to the attractiveness, perspicuity, efficiency, stimulation, and novelty of the IoTutor. The evaluation results show a trend that students express an overall positive attitude towards the IoTutor with majority of the aspects rated higher than the neutral value.

Keywords: Internet of Things (IoT) · education · cognitive computing · IBM Watson

1 Introduction

Internet of Things (IoT) [17] pertains to networked interactive physical objects (such as, personal devices, connected cars, industrial machines, or household goods) with sensing, processing, communication, and acting capabilities. Evolution of the Internet from a network of computers to the network of things creates opportunities for new services and applications in society [12, 21, 24] and industry [9, 6]. According to Gartner [14], it is expected that by the year 2020 there will be about 20 billion IoT devices worldwide and it is expected that more than 65% of companies will use IoT solutions. Therefore, adequate education of the future workforce is a precondition for success in the increasingly relevant domain of IoT.

When searching for information about a topic, search engines usually return more results than we can study. For instance, currently Google returns about 35

* This research has received funding from the Swedish Knowledge Foundation under Grants No. 20150088 and No. 20150259

million results, when we search for *Internet of Things*. Therefore, it is important to have a system that returns succinct information from relevant literature based on a question expressed in natural language.

A cognitive computing [8, 19, 23] system, such as the IBM Watson [20], relates a text passage (that is a question) with another text passage (that is an anticipated corresponding answer) by using machine learning. Predicting the probable answer involves determining the major features of the question by generating hypothesis and evaluation of possible answers considering the context, and iterative learning from each instance of interaction with the cognitive system. The IBM Watson [13] has been successfully used in many domains [22], such as, the life sciences research [10] or health-care [5]. Goel et al. [15] argue that the IBM Watson has the potential to be used as an educational tool.

In this paper, we propose to use cognitive computing for education of students in the IoT domain. We describe the design and implementation of IoTutor that we use for empirical evaluation of our approach. We have implemented the IoTutor as a platform-independent web-based application using a collection of the IBM Watson cloud services including the discovery service, text-to-speech and speech-to-text services. We trained the IoTutor with selected scientific publications and course books relevant to the IoT education. To investigate users' experience with the IoTutor, we asked a group of students taking an IoT master level course at the Linnaeus University in Sweden to use the IoTutor for a period of two weeks. One of the course assignments was to develop a literature review for the course project. To complete the assignment, they were instructed and encouraged to use IoTutor beside Google Scholar and other digital libraries. Via a user experience questionnaire participants were asked to express their opinions with respect to the attractiveness, perspicuity, efficiency, stimulation, and novelty of the IoTutor. The results show a trend that participants expressed an overall positive attitude towards the tool. The majority of aspects were rated higher than the neutral value, while the rest were slightly lower than the neutral value. We observed that for some questions IoTutor showed sub-optimal answers, which may be a consequence of using a relatively small number of training questions and papers.

Major contributions of this paper include,

1. a development of IoTutor, which is a cognitive computing tool able to interact with users through text and voice using natural language,
2. a training of IoTutor for education of students in the IoT domain,
3. an evaluation of IoTutor with the help of a group of students at Linnaeus University.

The rest of the paper is organized as follows. In Section 2 we discuss the related work. Section 3 describes the design and implementation of IoTutor. In Section 4 we first demonstrate the use of IoTutor, and thereafter we discuss the results of the user experience questionnaire. We conclude our paper and provide future research directions in Section 5.

2 Related Work

In this section we first provide examples of related work and thereafter we contrast the work presented in this paper with the related work.

Goel et al. [15] used different Watson services to develop six diverse applications that aim at understanding the functionality and capabilities of Watson and enhancing the human-computer co-creativity. By developing these diverse applications the authors argue that Watson has potential to be used in different domains, and has large range of opportunities to be used as an educational tool.

Achilleas et al. [4] propose to use social networks and media for increasing motivation of students for STEM education and careers. They developed a social media aware platform and evaluated it in the context of a pan-European contest in STEM disciplines. More than 700 pre-university students participated in the contest and via a user-experience questionnaire they had the opportunity to express their opinion. Results of the questionnaire suggest that the contest using social media had positive influence on learning and motivation of participants.

Chen et al. [10] investigate the use of IBM Watson for accelerating the life sciences research. The authors trained Watson using large amounts of data including pharmacological data, genomics data, patents, and literature in life sciences. Watson is able to recognize concepts and their synonyms when they appear as an image or text in literature. For instance, Watson was able to generate in real-time relationships between the multiple sclerosis and any gene using data from more than 26 million MEDLINE abstracts.

Witte et al. [26] uses natural language processing to bring new levels of support to software developers. A plug-in that is integrated in the Eclipse IDE is developed, which provides quality analysis of the comments found in source code and version control commits. The aim of this project is to help software developers reduce the effort required to analyze their code by extracting useful information that might be valuable to understand the functionality of the application that is not always obvious by looking at the source code only.

Chozas et al. [11] study the use of cognitive computing for assisting novice programmers in avoiding the commonly made mistakes in parallel programming with OpenMP. They use the dialogue service of the IBM Watson for implementation of their solution that enables a dialog-based interaction with a programmer in English and Spanish during the process of parallel programming.

Harms [18] proposes an approach that is able to monitor and understand the programming skill level of the developer and adaptively suggest code examples that may help to learn new programming concepts found within the suggested examples. The author argues that this approach avoids overwhelming the memory of novice programmers by considering the previous knowledge of the programmer and carefully suggesting examples that contain new information.

In contrast to the related work, we use the cognitive computing technology to develop the IoTutor that assists students to learn about the domain of Internet of Things.

3 Design and Implementation

In this section, first we describe the design of our solution, thereafter we highlight implementation details.

3.1 Design

The goal of IoTutor is to provide means for communication (related to the area of Internet of Things) between the user and the computer through natural language in a similar fashion as personal assistants like Apple Siri, Google Now, and Microsoft Cortana. While these personal assistants are incorporated in the operating system, and can be used only within a selected operating system, we aim at developing a web based platform that is independent from the operating system and device. IoTutor allows users to interact with it, that is ask questions related to Internet of Things, through a dialog-based interface.

A high-level architecture overview of our system is depicted in Figure 1. The main components of the system are the front-end, the back-end, and the Watson Cloud Services. The user interacts with the front-end. The front-end forwards requests from the user to the back-end, and returns responses from the back-end to the user. The back-end is connected to Watson Cloud services, which are used to extract knowledge from a corpus of data (in this case scientific publications), and enhance IoTutor with speech capabilities (that is text-to-speech and speech-to-text).

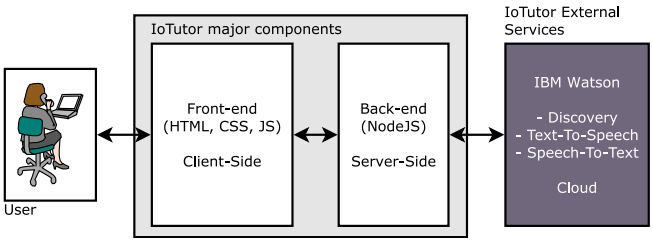


Fig. 1: A high-level overview of IoTutor architecture. The front-end provides means for interaction with the user, and the back-end interacts with the IBM Watson Cloud services. The front-end is a web based dialog view that can be accessed with any Internet Browser.

Figure 2 depicts the flowchart with the major events of the IoTutor. When the tool starts, the page is rendered and a welcome message is displayed. Then depending on how the user wants to interact with IoTutor, that is voice or text, the following steps are performed. If the user wants to use voice commands, then the microphone button should be pressed, which will establish a stream through the back-end between IoTutor and the speech-to-text Watson, and the recognized words will be displayed in the question box. Unless the user toggles

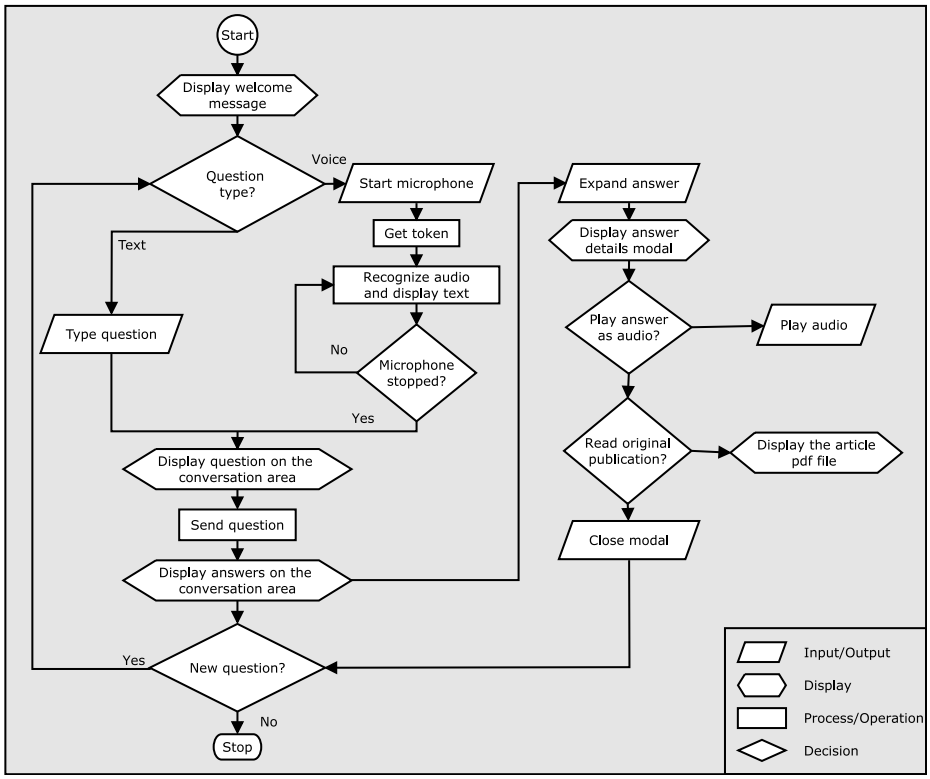


Fig. 2: A flowchart of user interaction with the IoTutor.

the microphone from on to off, IoTutor will keep recognizing the voice commands and display the text on the question box. Otherwise, if the user decides to use text to ask questions, the question can be typed on the question box. When the microphone is stopped, or the send button is pressed, the question will be displayed in the conversation area, and then it will be sent to the back-end. When the list of relevant answers is retrieved, those will be displayed in the conversation area. Passages of answers are displayed first, which the user may click to expand and then the answer will appear in a modal (pop-up) window with more details. Those details include a link to the full article and an option to let IoTutor read the full text to the user. Once the button to read the text is pressed, an audio player will be shown and IoTutor will start reading the text. If the link to the full article is pressed, the user will be redirected to the full pdf file. When the close button is clicked, the modal window will be closed, and the user may either choose to expand another answer, or ask a new question.

A high-level overview of the development process of IoTutor is depicted in Figure 3. There are four main activities, including data preparation, data import, training of the model, and using the model. In what follows we describe each activity.

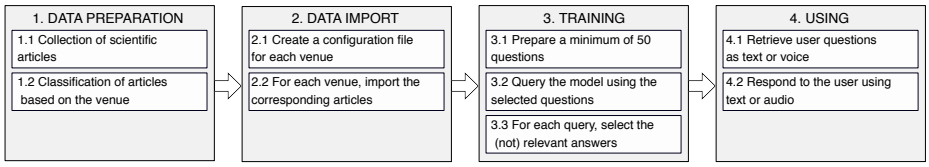


Fig. 3: IoTutor development process. Major steps include data preparation, data importing, IoTutor training, IoTutor using.

Data Preparation (activity 1): Scientific articles related to Internet of Things were collected from different electronic databases, including ACM, Springer, IEEE Explore, and Elsevier (activity 1.1). They were classified in respective folders, where each folder corresponds to an electronic database. The scientific articles were further classified by the venue, such as a conference, journal, or a magazine (activity 1.2).

Data Import (activity 2): To be able to split the scientific publications in subsections, which is useful to correctly identify all relevant sections of the manuscript, a separate configuration file was created for each database and venue. For example, a configuration file, named *Springer-conference-configuration.json*, was used to identify sections of Springer conference scientific articles (activity 2.1). These configuration files were used to import the data into the Watson Discovery service. The corresponding configuration files were used to import the scientific articles collected from each venue of a digital library (activity 2.2).

Training (activity 3): According to the Watson Discovery Service documentation [1], a minimum of 49 queries should be used to train the Watson Discovery service. We used minimal resources and have defined 50 questions to train the model (activity 3.1). For each question, we have added a natural language query to the Watson Discovery service for training (activity 3.2). For each question, Watson suggests a set of answers, which need to be marked as relevant or not-relevant (activity 3.3). We went through 15 answers for each question and marked their relevance. Since the training process is a one-time activity, we have used the Watson Discovery Tooling interface, rather than using the API to implement the same functionality. In total, we fed Watson with 50 paper and 2 books in the topics of Internet of Things. The training process took 6-7 hours excluding the time to find the articles and preparing the questions.

Using the service (activity 4): Once the Discovery service was trained, the tool was ready to accept various questions related to Internet of Things (activity 4.1). A set of relevant answers are provided by Watson, and displayed to the user (activity 4.2).

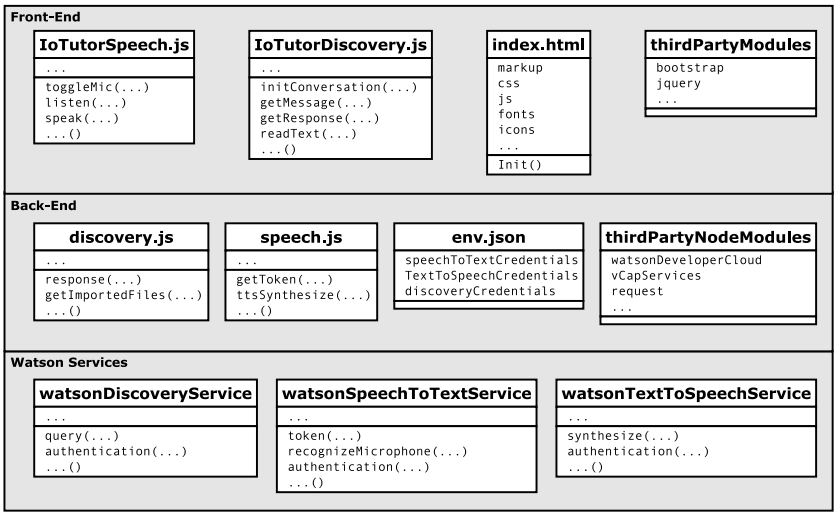


Fig. 4: An overview of components used to implement our solution.

3.2 Implementation Details

Figure 4 shows an overview of components used to implement our solution. There are three layers of the architecture, the front-end, the back-end, and the Watson Cloud Services.

Front-End The major components of the front-end include the *IoTutorSpeech*, *IoTutorDiscovery*, and *index*.

The *IoTutorSpeech* component has three main functions, *toggleMic()* used to turn the microphone on and off, *listen()* used to initiate the process of streaming data to the corresponding Watson services when the microphone is on, and *speak()* will start reading the corresponding answer when the user clicks the play button.

The *IoTutorDiscovery* component has four main functions, *initConversation()* used to initiate the conversation, which basically says the welcome message and some instructions on how to use IoTutor. The *getMessage()* is triggered when the user asks a question, it displays the question on the conversation area, and sends it to the corresponding Watson services. The *getResponse()* is triggered when the back-end has found a response and it displays it on the conversation area. The *readText()* will simply call the *speak()* function from the *IoTutorSpeech* component with the corresponding parameters.

The *index* component is a simple file which includes the html markup, the style-sheets, java-scripts, fonts, and icons. It also has an *init()* function which is used as a constructor to set the variables. Third party libraries such as bootstrap and jQuery were used to implement the front-end.

Back-End The major components of the back-end include the *discovery*, *speech*, and *env*. Third party NodeJS modules, such as Watson Developer Cloud SDK, vCapServices and request, were used in our implementation. For design and security reasons, the front-end communicates with the Watson services through the back-end. The back-end has the environment file (*env.json*) that contains the credentials (such as, username, password, url, workspace, version, collection_id, configuration_id, and environment_id) for each of the Watson Cloud services, including discovery, text-to-speech, and speech-to-text.

The *discovery* component is a simple application program interface (API), which accepts requests (in this case questions) from the front-end and sends it to the Watson Cloud Discovery service. The API needs to authenticate first using the information found in the *env.json* file. When a response is received from the Watson Discovery service, the back-end forwards the response to the front-end. Additionally, the back-end has a database of files that were imported in the Watson Discovery service, and it can easily map a response to an actual scientific publication, such that if the user wants to read more, the front-end can provide a link to the paper.

The *speech* component is an API, which handles requests and responses for speech-to-text and text-to-speech services. It can basically establish a stream that can listen to the user's microphone and display the recognized text in the input box of the IoTutor GUI, as well as can generate an audio file corresponding to a given input text. Similar to the *discovery* component, it first authenticates to the text-to-speech or speech-to-text service using the information provided by the *env.json* file, and then it can send specific requests to the Watson Cloud text-to-speech or speech-to-text services.

Watson Cloud Services Watson provides different cloud services, such as speech-to-text, text-to-speech, discovery, conversation, and natural language understanding. To achieve the goals of our paper, we have used only three of them, *discovery*, *text-to-speech*, and *speech-to-text*.

The *discovery* service allowed us to extract useful information from various scientific publications related to Internet of Things, such that when the user asks a question, we can query the service and retrieve a list of ranked responses (publications, sections of publications, or a specific sentence or paragraph in such articles) that are relevant to the question being asked.

The *speech-to-text* service allowed us to enhance the IoTutor with voice recognition, such that the user may use their microphone to ask questions. This service will listen to the microphone and as a response will provide a stream of recognized text.

The *text-to-speech* service allowed us to enhance the IoTutor with the possibility to read the provided answers for users. This service accepts a text input and provides an audio file which can be played on demand by the user. The combination of the *speech-to-text* and *text-to-speech* services enabled us to provide an interaction between the user and IoTutor, similar to personal assistants like Apple's Siri [7] or Google's Now [16].

Environmental details To implement our application, we have used HTML5, CSS, and JS in front-end, whereas NodeJS [2] is used in the back-end. Among others, in the back-end we used the Watson Node SDK [3] to access the IBM Watson Developer Cloud services. The application was deployed on an Ubuntu v14.04 server with Apache v2.4 and node v6.11 installed. For the voice commands to work on the front-end, which requires data to be encrypted, we enabled the Hyper Text Transfer Protocol Secure (HTTPS) on our server (Table 1).

Table 1: Key software components of IoTutor server

Software component	Version
Operating System	Ubuntu 14.04 x64
Web Server	Apache 2.4
Runtime Platform	NodeJS 6.11
Libraries and Packages	Watson Node SDK 2.4, Unirest 0.51, Bootstrap 3.3.7

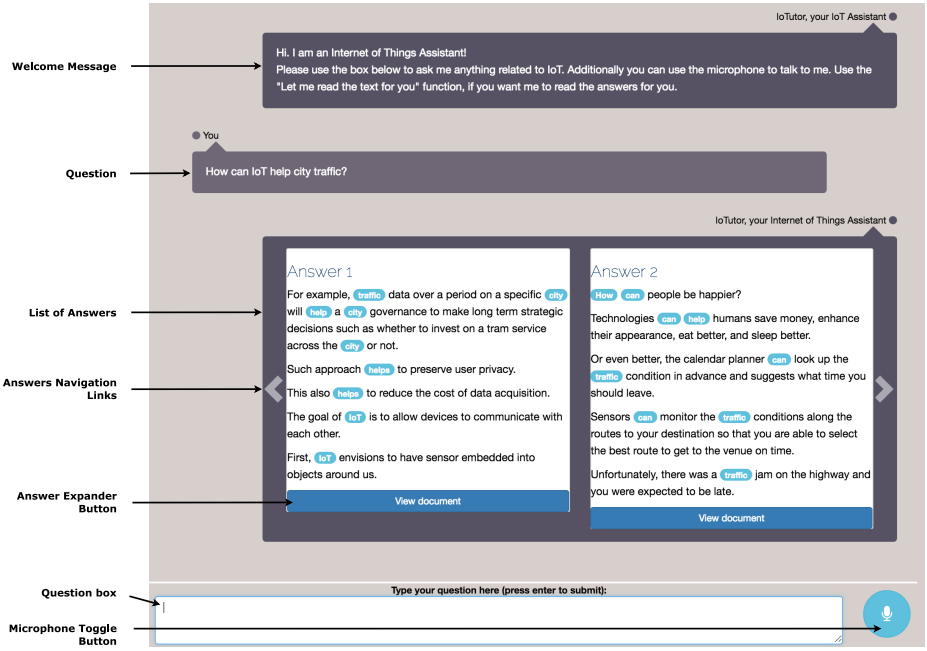
4 Evaluation

In this section, we first demonstrate the functionality of IoTutor and how users can interact with IoTutor to ask questions related to Internet of Things. Thereafter, we describe the evaluation method and results of our evaluation.

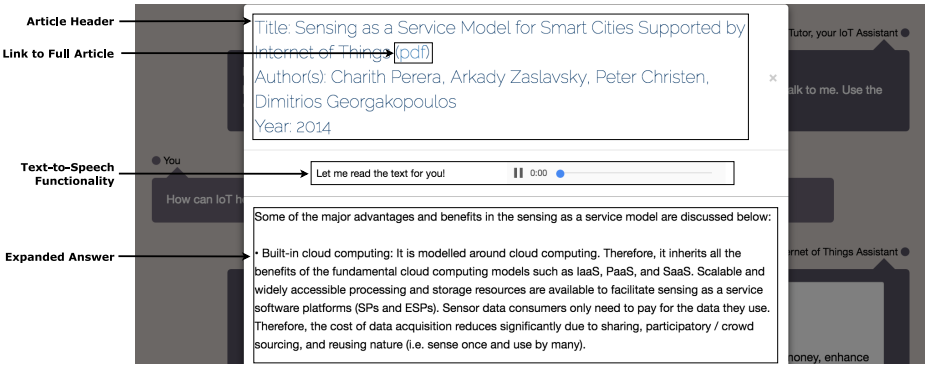
4.1 Demonstration

Figure 5 shows the graphical user interface of IoTutor and demonstrates a use case scenario when a user asks an IoT related question, and IoTutor provides a list of relevant answers. When IoTutor is loaded, the welcome message is displayed (see Fig. 5a) and waits for the user to either press the microphone button and talk, or type a question on the question box. The question will appear in the conversation area together with the list of relevant answers. IoTutor provides up to 10 relevant answers, out of which the first two will appear first, and the rest can be accessed using the navigation links. To help users quickly find a desired answer, the list of answers shows only excerpts of the full answer, which can be displayed when the user clicks the *View Document* button.

The *Expanded answer view* is depicted in Fig. 5b, and it contains information related to the scientific publication that contains the answer, including article title, authors, and publication year, as well as a link to the full article. The text shown in this view provides the complete information, which sometimes was lengthy, and to help the user focus on the most important parts, we highlight the passages in a light blue color. Instead of reading the text, the user may choose to let IoTutor read the text. In that case, an audio player will be shown containing the controls to play, pause, or move forward and backwards through the audio stream.



(a) The conversation area view, including the greeting message, questions, list of answers, question box, and microphone toggle button.



(b) The expanded answer view, including the article information (title, authors, year), the text-to-speech functionality, and expanded answer.

Fig. 5: Demonstration of a case where the user: (1) asks a question and IoTutor shows a list of answers (Figure 5a); and (2) clicks the *View Document* button to expand the answer (Figure 5b).

Table 2: The user experience questionnaire for the evaluation of IoTutor.

Aspect	Id	Question
Attractiveness	a1	What is your overall impression of our interactive Internet of Things Assistant (IoTutor)?
	a2	How useful do you find the possibility to ask questions using voice?
	a3	How useful do you find the feature that allows IoTutor to read the answers for you?
	a4	How useful do you find the functionality of IoTutor that returns research articles as answers to some of the questions?
Perspicuity	p1	How intuitive and easy to understand is the GUI (Graphical User Interface) of IoTutor?
	p2	How difficult is to get familiar with IoTutor?
	p3	How easy it is to use IoTutor?
Efficiency	e1	How efficient is IoTutor to help you find answers for questions related to Internet of Things?
	e2	How quickly did IoTutor find the answers?
Stimulation	s1	How valuable is to use IoTutor for the assignment?
	s2	How exciting is to use IoTutor for the assignment?
	s3	How interesting is to use IoTutor for the assignment?
	s4	How much does IoTutor motivate you to learn about the Internet of Things?
Novelty	n1	Dull / Creative
	n2	Conventional / Inventive
	n3	Usual / Leading edge
	n4	Conservative / Innovative

4.2 Evaluation Method and Results

To understand users' interaction and experience with the IoTutor, we conducted an evaluation of the tool. The IoTutor was initially presented to participants and then they were given two weeks period to explore it. Most of the participants were students taking an Internet of Things master level course at Linnaeus University. One of the course assignments was to develop a literature review for the course project. To complete the assignment, they were instructed and encouraged to use IoTutor beside Google Scholar and other scientific libraries. After they had used IoTutor, participants were instructed to answer questions (see Table 2) of a User Experience Questionnaire (UEQ), which was adopted from [25]. The questionnaire measured six dimensions: *attractiveness* (four questions), *perspicuity* (three questions), *efficiency* (two questions), *stimulation* (four questions), and *novelty* (four questions). Questions were represented in a five-scale semantic differentials, where 1 indicates the most negative answer, 5 indicates the most positive answer, and 3 indicates a neutral answer. Ten participants consented and answered the questionnaire.

The results of the UEQ show a trend that participants expressed an overall positive attitude towards the tool. Most of the aspects/dimensions (12 out of 17) were rated higher than the neutral value, while the rest were slightly lower than the neutral value (see Figure 6). However, the highest average rating was 3.8, which is an indication that although positive, these results are still not convincing.

When looking cumulatively at each of the five aspects/dimensions, all ratings appear over the neutral value (see Figure 7). Considering that the aspects of *perspicuity* and *efficiency* measure the usability of the tool, and the aspects of *simulation* and *novelty* measure the user experience, the indication is that both show similar ratings. Slightly higher ratings were shown for *perspicuity* and *novelty*, which is an indication that participants had no difficulty to familiarize themselves with using the tool, and participants had recognized the innovativeness and creativeness of the tool.

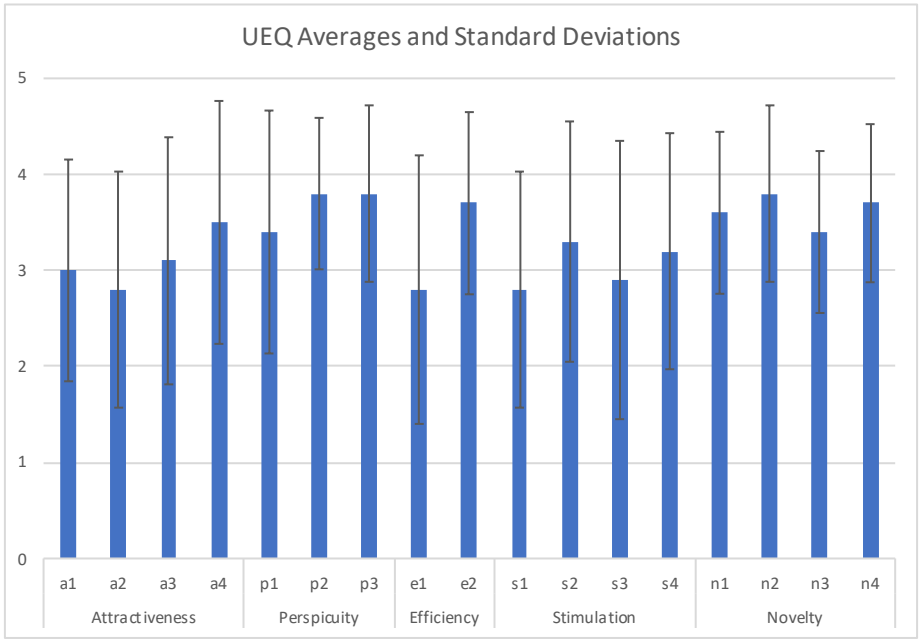


Fig. 6: The average and standard deviation for each of the answers in the user experience questionnaire.

Besides the UEQ results, we had an opportunity to briefly discuss the tool with two participants after they have used the tool and provided their UEQ answers. One major concern expressed was that the tool provided keyword-based answers, which was not expected considering that the interface tool was accepting natural questions. Participants' expectation was that an interface that

is based on Watson should provide more natural and comprehensive answers. Indeed, such interface behavior was also noted by the researchers. Because this seemed inappropriate, we did contact IBM and inquire whether the service was working correctly or if training process and resources fed to Watson were with omission. Their reply confirmed that the procedure we followed was correct. It remains to speculate that perhaps in order to get better results, Watson should be trained with more content and more questions than what we had provided (described in activity 3).

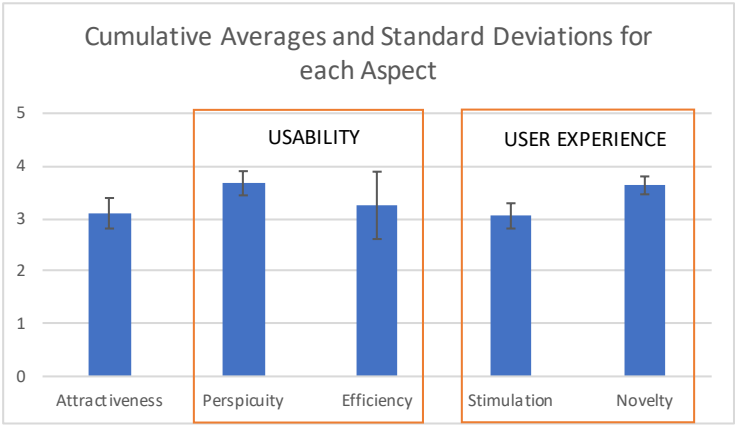


Fig. 7: Cumulative averages and standard deviation for each aspect of UEQ.

Two other usability issues were revealed in our discussion. One was considering the voice-based input. Apparently, when users used that feature, once they stopped talking, the question was submitted, without offering the opportunity to edit the text beforehand. Considering that sometimes there were comprehension issues and the text displayed by the interface was different from what the participant had uttered, being able to edit the question was necessary. The other issue was the interface feature to read out loud the excerpts from the paper provided as an answer. This feature was actually possible only when participants expanded the initial answer and viewed an extended excerpt from the source paper. Participants expected that the interface would read even the initial brief answer provided. This was an interface omission to clearly communicate when that feature is availability.

5 Conclusions and Future Work

With above 20 billion IoT devices expected to be used in the near future, adequate education of skilled work-force is a priority. Considering the recent success

of cognitive computing solutions in health or financial domains, and the increasing volume of IoT literature in various formats, we have proposed in this paper the application of cognitive computing to IoT education.

We have described the design and implementation of IoTutor, which is a platform-independent web-based application that enables dialog-based interaction with users that want to learn about IoT. The interaction in text or speech form is done using natural language. We have investigated the usefulness of IoTutor by asking a group of students at the Linnaeus University to use the IoTutor for a period of two weeks. Participants have expressed their opinions with respect to the attractiveness, perspicuity, efficiency, stimulation, and novelty of the IoTutor. The majority of aspects were rated higher than the neutral value, while the rest were slightly lower than the neutral value.

We have observed that using a relatively small number of training questions and papers may result with sub-optimal answers from IoTutor. Our intention with this study was to see the quality of answers that IoTutor would provide with minimal training resources. In the future, we plan to use a larger digital library of scientific publications during the training process of IoTutor to measure its impact in the increase of the quality of the answers provided by the tool.

References

1. IBM Cloud Docs: Discovery - Improving result relevance with the tooling, https://console.bluemix.net/docs/services/discovery/index.html, (accessed Apr. 17, 2018)
2. Node.js, https://nodejs.org/en/, Node.js Foundation, (accessed Mar. 4, 2018)
3. Watson Developer Cloud Node.js SDK, https://github.com/watson-developer-cloud/node-sdk, IBM Open Source at GitHub (accessed Feb. 15, 2018)
4. Achilleos, A., Mettouris, C., Yeratziotis, A., Papadopoulos, G., Pllana, S., Huber, F., Jaeger, B., Leitner, P., Ocsovszky, Z., Dinnyes, A.: SciChallenge: A Social Media Aware Platform for Contest-Based STEM Education and Motivation of Young Students. IEEE Transactions on Learning Technologies (March 2018). https://doi.org/10.1109/TLT.2018.2810879
5. Ahmed, M.N., Toor, A.S., O'Neil, K., Friedland, D.: Cognitive Computing and the Future of Health Care Cognitive Computing and the Future of Healthcare: The Cognitive Power of IBM Watson Has the Potential to Transform Global Personalized Medicine. IEEE Pulse **8**(3), 4–9 (May 2017)
6. Alsouda, Y., Pllana, S., Kurti, A.: A Machine Learning Driven IoT Solution for Noise Classification in Smart Cities. In: Proceedings of the Machine Learning Driven Technologies and Architectures for Intelligent Internet of Things (ML-IoT). pp. 4–9. ML-IoT, Euromicro (2018)
7. Apple Inc: iOS - Siri - Apple. http://www.apple.com/ios/siri/ (2017), online; accessed Feb. 9, 2018
8. Cer, D., Yang, Y., Kong, S.y., Hua, N., Limtiaco, N., St. John, R., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y.H., Strope, B., Kurzweil, R.: Universal Sentence Encoder. ArXiv e-prints (Mar 2018)
9. Cerf, V., Senges, M.: Taking the Internet to the Next Physical Level. Computer **49**(2), 80–86 (Feb 2016). https://doi.org/10.1109/MC.2016.51

10. Chen, Y., Argentinis, J.E., Weber, G.: IBM Watson: How Cognitive Computing Can Be Applied to Big Data Challenges in Life Sciences Research. Clinical Therapeutics **38**(4), 688 – 701 (2016)
11. Chozas, A.C., Memeti, S., Pllana, S.: Using cognitive computing for learning parallel programming: An IBM Watson solution. Procedia Computer Science **108**(Supplement C), 2121 – 2130 (2017)
12. Ferati, M., Kurti, A., Vogel, B., Raufi, B.: Augmenting Requirements Gathering for People with Special Needs Using IoT: A Position Paper. In: Proceedings of the 9th International Workshop on Cooperative and Human Aspects of Software Engineering. pp. 48–51. CHASE '16, ACM, New York, NY, USA (2016)
13. Ferrucci, D.A.: Introduction to "This is Watson". IBM Journal of Research and Development **56**(3.4), 1:1–1:15 (May 2012)
14. Gartner: Internet of Things, https://www.gartner.com/technology/research/internet-of-things/, (accessed Mar. 3, 2018)
15. Goel, A., Creeden, B., Kumble, M., Salunke, S., Shetty, A., Wiltgen, B.: Using Watson for Enhancing Human-Computer Co-Creativity. In: 2015 AAAI Fall Symposium Series (2015)
16. Google Inc: Google Now. https://support.google.com/websearch/answer/4541722 (2017), online; accessed Feb. 9 2018
17. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems **29**(7), 1645 – 1660 (2013)
18. Harms, K.J.: Towards a programming environment that adaptively suggests examples and corresponding puzzles based on programmer skill. In: 2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). pp. 185–186 (July 2014). https://doi.org/10.1109/VLHCC.2014.6883047
19. Henderson, M., Al-Rfou, R., Strope, B., Sung, Y.h., Lukacs, L., Guo, R., Kumar, S., Miklos, B., Kurzweil, R.: Efficient Natural Language Response Suggestion for Smart Reply. ArXiv e-prints (May 2017)
20. High, R.: The Era of Cognitive Systems: An Inside Look at IBM Watson and How it Works, http://www.redbooks.ibm.com/redpapers/pdfs/redp4955.pdf, REDP-4955-00, Dec. 12, 2012 (accessed Mar. 3, 2018)
21. Memedi, M., Tshering, G., Fogelberg, M., Jusufi, I., Kolkowska, E., Klein, G.: An Interface for IoT: Feeding Back Health-Related Data to Parkinson's Disease Patients. Journal of Sensor and Actuator Networks **7**(1) (2018)
22. Mercer, C.: 16 innovative businesses using IBM Watson: Which companies are using Watson's big data and analytics to power their business? http://www.computerworlduk.com/galleries/it-vendors/16-innovative-ways-companies-are-using-ibm-watson-3585847/ (2017), online; accessed Mar. 12 2018
23. Modha, D.S., Ananthanarayanan, R., Esser, S.K., Ndirango, A., Sherbondy, A.J., Singh, R.: Cognitive Computing. Commun. ACM **54**(8), 62–71 (Aug 2011)
24. Perez, D., Memeti, S., Pllana, S.: A simulation study of a smart living IoT solution for remote elderly care. In: 2018 Third International Conference on Fog and Mobile Edge Computing (FMEC). pp. 227–232 (April 2018)
25. Schrepp, M.: User Experience Questionnaire Handbook. All you need to know to apply the UEQ successfully in your project (2015)
26. Witte, R., Sateli, B., Khamis, N., Rilling, J.: Intelligent Software Development Environments: Integrating Natural Language Processing with the Eclipse Platform, pp. 408–419. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)