ISSN 2278-3091
Volume 9 No.2, March - April 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse134922020.pdf
https://doi.org/10.30534/ijatcse/2020/134922020

Improving the Correctness of Chatbots using Multi-Response-Based Aggregation for Big-Data

Umme Ayeman Khan[1], Dr. Ramchand Hablani[2], Sarang Jain[3]
Shri Ramdeobaba College of Engineering & Management,Nagpur-440013
Maharashtra, India, code wizards Technologies Manish Nagar Somalwada, Nagpur
Email:khanua_1@rknec.edu,hablanir@rknec.edu,codewizardstech@gmail.com

ABSTRACT

Currently the communication systems we have in real life has very little activity and a very little interaction among its resident avatar bots and smart objects. Chatbots can resolve this issue by adding faster and more interactive interfaces to the end product. But, sometimes chatbot responses can be slow and limited. This paper aims to overcome this defect by introducing machine learning entities into the chatbots. Our goal is to create an architecture which can be attached to any object in real life and which will cause the object to immediately become a chatting object. So far, we have found that we need to work with existing chat-bot implementations and modify it to be useful in the real-world setting and then find a way to link the new chat-bot into other systems. Furthermore, this work will be improving the responsiveness of the implemented chatbot using data aggregation techniques.Which results in a reduced delay, and improved user experience. This work is directed towards college-based chatbots system, wherein student queries can be resolved in real-time with the help of aggregated data. The proposed chatbot system will be utilizing natural language processing in order to find out action words, and then perform matching using Jaccard distance in order to evaluate the best matching responses. Moreover, Jaccard distance will also be used against dynamic datasets in order to evaluate dynamic responses to user queries. The result and analysis of the algorithm indicate that the proposed algorithm is faster and more effective than single-query based systems.

Key words: Accuracy, aggregation, chatbots, machine learning.

1. INTRODUCTION

In the future, where “everything is alive” (EiA), all objects will have identity, an ability to communicate, and a way to interact with other objects and humans. Without the ability to interact through spoken, or written, language the objects in the hospital will not be able to effectively communicate in an EiA setting. They can, of course, communicate through other means via the 'magic' of computing but this will not play well to the EiA idea. In a real hospital or any real human setting there are people who talk. Our main form of communication is the spoken language and as a result we want to see entities in the hospital communicating.

This provides many challenges as modern artificial intelligence has not yet 'solved' the spoken or written language understanding problem. Repeating a script is simple, but getting objects to interact in a lifelike matter is not. Chat bots need excellent natural language processing to understand what is being asked and to respond intelligently. This two-sided coin presents many problems for programmers and it has been worked on since the early 60's with only moderate success. The objective of the “Everything is Alive” project at the University of Arkansas is to create a world where everything is both interconnected and interactive. The inter-connectivity will have advantages such as being able to do just about anything from just about anywhere. The interactivity will have advantages of being able to get just about any information from just about anywhere. These of course are only two examples of the advantages of a world where every object can communicate both with humans and with other objects [1].

At the moment pervasive computing is an idea not fully realized. With 3D virtual worlds it is easier to get an idea of what pervasive computing may one day mean. With simulations such as the hospital we can begin to see what exactly will happen when “Everything is Alive.” What is the full potential of such a scenario? What disadvantages are there? Using virtual worlds, we are able to get an idea about these things before they are here. With chat bots in the virtual world we are hoping to get an idea of how things will interact once both objects and humans in the virtual world can talk to one another. As spoken language is the primary form of communication human to human it is very important to see how, in a pervasive computing environment, the ability for objects and humans to talk to each other would play out. Chat bots hold endless possibilities to solving everyday lives both in the virtual world and in the real world [9]. A chat bot in the Second Life world of the University of Arkansas might work for the virtual hospital and help give advice to patients without the use of an actual real life doctor. The same could be said

for the real world. Chat bots might also help keep people company by giving someone something to talk to and carry out a conversation with. Currently chat bots hold a major impact already with customer service departments within companies as they help limit the flow of customer problems to what might be a short-staffed customer service line keeping both the employees happy with fewer calls and the callers happy with shorter wait times. Another potential impact of our project is aimed to helping build a chat bot to help speech pathologists in training learn how to speak to people with a speech disability. In order to do this, however, special speech recognition software would need to be implemented to read in the trainee's speech to text as well as translate the computer's test output to speech with disability [8]. With a growing computing community, it would be a shame not to take advantage of Artificial Intelligence in the chatter bot realm to help our daily dilemmas, especially those already mentioned.The next section describes various chatbot systems& their nuances, followed by the proposed algorithm, and finally the result evaluation of the proposed algorithm on various input conditions. We conclude the paper with some interesting observations about the proposed chatbot, and some further research which can be done in order to extend this work.

2. **LITERATURE REVIEW**

The main related technologies are previously developed chat bots such as Eliza [3], Parry [4], A.L.I.C.E. [5] and Jabberwocky [6]. The short comings of these technologies are all the same. They simply cannot yet pass a Turing test unless the user queries are very narrowly defined. For the most part these technologies are based on pattern matching and have very little or no reasoning involved. Jabberwocky is the sole exception to this as it can 'learn' as you speak with it. It does this by storing all user interactions with it and attempting to find more appropriate responses .Other technologies include natural language processing which is a main off shoot of artificial intelligence. In order to get chat bots working properly they will need the ability to in some way understand what humans are saying. Virtual worlds provide a place where we can test out the chat bots and see how they interact. Pervasive computing will find chat bots useful in that they will provide an incredible way for objects which are alive to communicate both with each other and humans .Many chat bots have already been created and developed already. Most chat bots these days use a chatterbot brain that relates back to Eliza, Parry, Alice or Jabberwocky as they are some of the most advanced bots programmed to this date and provide for a structural basis for teaching new bots how to talk. Also, see key Technologies for Eliza [3], Parry [4], Alice [5], and Jabberwocky [6] in references .Robust Sentence Analysis and Habitability, A thesis concerning the topics of natural language processing and the habitability problem. [7]

The iPhone now uses voice recognition software. [8] "Mybotai" is an AOL Instant Messenger screen name that is actually a bot. When talking to mybotai it almost seems as if you are talking to a real person. It is the best bot I have encountered to this day and is the learning summation of many years of work. When talking to him I couldn't even find out what kind of chatterbot brain he had because he stated he was better than the rest. Some of the recent projects undertaken using chatbots are,

- Mirror Worlds project – this project develops a script which can be used to make objects in the mirror world talk. This helps emulate the real world. However, with the time given in a semester and our limited knowledge of Action script and Linden Script we were unable to port our project into Second Life, leaving it for future work and development.
- Ontology project – this project develops a script which covers the hospital chat ontology. [10]
- Soft Controller project – using chatbots soft controllers could one day talk to their users making information exchange quicker and easier. [11]
- Smart Devices – similar to soft controllers our chat bout could one day be used to enable smart devices to talk to humans and to other smart devices. Along with controlling these devices by being asked to turn something on or off a smart device might also brag about its capabilities and let you know what it might be able to do. [12]
- Workflow – again a chat bot with good natural language processing could be used to quickly and effectively parse spoken commands and to communicate what needs to be done. [13]
- Search Spider – with a proper natural language processor a search spider could listen to conversations and parse important information. [14]

Games in SL – the ability of the game to talk to a person via our chat bot could be used to enable blind people to play games in Second Life and as with other areas it would make information exchange easier and quicker. The next section describes our chatbot implementation followed by its results.

3. **PROPOSED AGGREGATION-BASED CHATBOT**

The architecture of our project can be split up into a few main components that drive it. The first core component is the chat bot shell. It is made up of Flash Action Script that operates the read/respond functionality of the bot and operates its speech patterns based off of a XML sheet that educates the bot into what to say.

In order to access this Action Script you must open the bot ".fla" file in Adobe Flash (we used CS4 in the Union and J.B. Hunt GACL on the Macintosh machines). Once you open the ".fla" file you need to make sure that the first layer is selected (the actions layer) and then go to "window" in the top menu

and pull down the actions pane. Now you'll see an extensive list of code that runs and operates the bot.

The main education of the bot is hosted within the XML sheet. In order to educate the bot, you must make a parse tree for the bot to search through. For example, <parse1><parse2> and when you are ready to define the education within the parse words you must place before the education <answX> so that the code knows that this is an answer. The problem we realized with the parsing is that the code in the action script only allows for a bi-dimensional array meaning that the bot can only parse two words. A tri-dimensional array might prove to be more effective with the bot education as some questions are a bit vague with two key words instead of three. For examples of this you may look over our XML sheet and see which responses we wanted to parse more words into to make it a more effective communicator.

If a chat bot wanted to learn what it was talking about and reference it in the future our chat bot would need a new XML sheet and implement a new function inside of the Action Script that allows for the bot to input into the XML sheet any topics it might not know about so that when asked about the topic in the future it might have a clear understanding of what it is then saying. Another useful idea for educating a bot through the bot's own self-education just mentioned would be having a trainer for the bot. This would be someone who could talk to the bot just to educate it and trigger these learning functions inside of the bot. This would have to be someone that is trusted so that the bot does not grow corrupt in its own language. A learning bot is like a child. Its formal knowledge of communication is very primitive and growing. A learning chat bot will pick up new phrases the same way it was taught. If you teach a kid a word that is nasty or dirty then the kid might repeat this later. You are effectively doing the same with the bot.

The other core component of the chat bot would have been the Linden Script that enables any object or bot with the script running to listen for people chatting with it as well as drive responses from a server. The main concept in this project is to connect the script with the chat bot shell. The script can listen for active conversation but it cannot read and respond. The chat bot shell can communicate with people but not without a main driving interface that listens for active conversation. In order to design these bots the Linden Script and Action Script will have to be ported so that the shell accepts incoming language from the game as well as decipher and rebound the response in an outgoing port back to the game. The linden script will listen for the text and pass it along to the server rather than try to read and respond by itself.

With hosting the actual chat bot on a server, it will make it possible to drop the linden script on any object that we desire to have it talk about healthcare to us. The difference between our chat bot and a chat bot that only knows general things (such as a smart controller) is that our chat bot is fairly extensively educated with over fifty medical responses that have been added to its XML sheet as well as being able to make reasonable assumptions about your symptoms to draw up what might be wrong with you.

The last thing about our chat bots is that we eventually wanted it to be able to or have the option to stutter so that it might assist others with learning how to communicate with people with a speech disability. Such problems with stuttering include repeated first consonant, repeated word, block, as well as any starting and stopping in the middle of a sentence. The chat bot would also need to have a text to speech interpreter that was programmed with these speech disabilities to make communication and education with a stutter bot more effective. The results and analysis of the integrated proposed algorithm is given in the next section.

4. PROPOSED CHAT BOT DESIGN

The block-diagram of the proposed chat bot can be seen from the following figure,

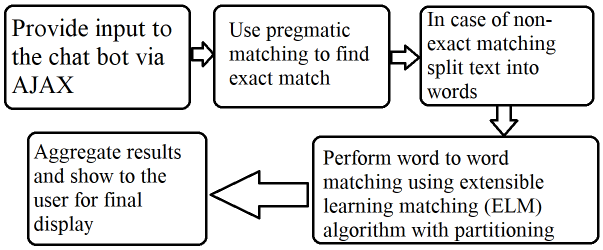

Figure 1: Block diagram of the proposed system

From the block-diagram we can verify that the input is first given to the matching engine via AJAX, which maintains a very good user experience for chatbots. These inputs are given to a pragmatic matching algorithm, which performs exact matching of the text with each of the lines at the input in order to provide the final result. Our system constructs object and relationship sets and constraints when users define form by making use of our basic patterns. Our system obtains object-set names from user-specified form titles and column labels. First of all, when users give the title to the base form of an application, our system takes the title as the name of the primary object set in our ontology. Then, when users add elements to the form by using one of patterns, our system generates object sets for each element by taking user-specified column labels as the object-set names. After constructing object sets, our system constructs relationship sets between these added object sets and the object set named by the form title, and then adds participation constraints on all object sets. The following example shows how our system constructs object and relationship sets and constraints. As shown in

Figure 3, assume that a user adds one element of each pattern in Figure 2 into a base form called "Base" with 3 rows for patterns (b) and (d) and 2 columns for patterns (d) and (e). Our system generates object and relationship sets and constraints as follows,

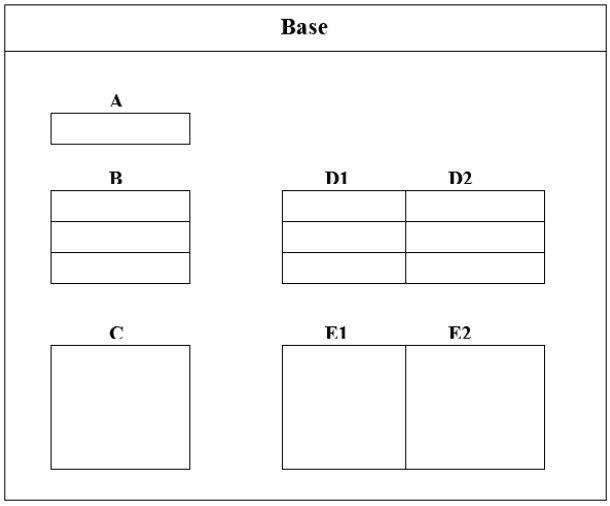

Figure 2: A User-Defined Single Form

Base [0:1] A [1:*] (a)
Base [0:3] B [1:*](b)
Base [0:*] C [1:*](c)
Base [0:3] D1 [1:*] D2 [1:*](d)
Base [0:*] E1 [1:*] E2 [1:*] (e)

where A, B, C, D1, D2, E1 and E2 are specified column labels for patterns (a), (b), (c), (d) and (e) respectively. Notice that our system generates binary relationship sets for the elements of single-column patterns (a), (b) and (c) and n-ary relationship sets for elements of multiple-column patterns (d) and (e). For object set Base representing the current form, the minimum participation constraint is 0 by default and the maximum of constraint is 1, 3 (a number specified by users) or * (an unlimited number). The constraints on the added object sets are always [1:*].

Every object set in a form corresponds to either a lexical or a non-lexical object set in our ontology. Object sets containing all string values correspond to lexical object sets, while object sets containing all nested forms correspond to non-lexical object sets. If a user nests a form in pattern (a), which has a single row or value, our system assigns the object-set name to be the title of the nested object set. For all other patterns, which have multiple rows, the object-set name with an appended row number becomes the title of the corresponding nested form by default. The user can specify a meaningful title for each nested form. Multiple-row nested forms actually represent specializations in our ontology. For example, if a user defines nested forms in the "base" form in Figure 4, our system constructs the following object and relationship sets and constraints:

Base [0:1] A [1:*]
Base [0:1] D [1:*]
A [0:1] B [1:*]
A [0:1] C [1:*]
D1, D2 : D
D1 [0:1] E[1:*]
D2 [0:1] F [1:*] G [1:*]

The following equation was used to perform Q-learning,

$$Q_i^{\pi}(s,a) - Q_i^{\pi'}(s,a) = \gamma \sum_{s'} p(s'|s,a) \left[\sum_{a'} \pi(a'|s') Q_i^{\pi}(s',a') - \sum_{a'} \pi'(a'|s') Q_i^{\pi'}(s',a') \right]$$

Where, **Q** is the learning factor, **s** is the input query, **a** is the dataset for matching, $\boldsymbol{\pi}$ is the Jaccard distance function, and other dashed values are the processed versions of the input and dataset.

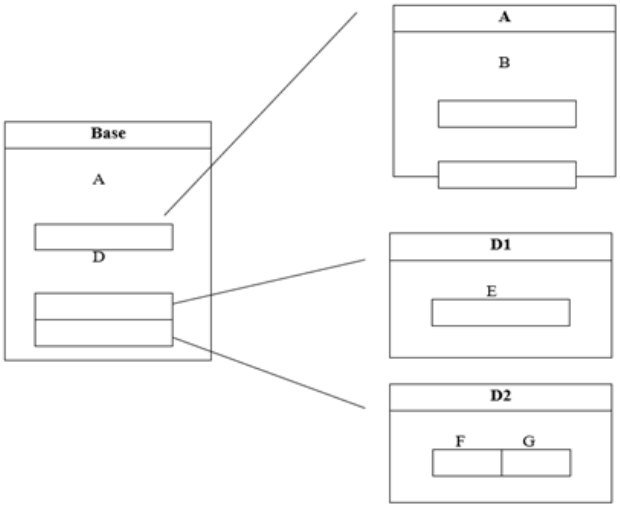

Figure 3: A User-Defined Nest Forms

Extraction patterns are regular expressions to describe data values. After a user defines a domain-dependent form and fills in it with values from several examples, our system tries to match these values against extraction patterns in the data-frame library. Sometimes there'll be just one extraction pattern within the library matching each object set. If several extraction patterns match, our system further makes use of name matching and context-and-keyword matching to select the most appropriate extraction pattern. If no extraction

pattern matches, our system provides a tool to help a user create a regular expression manually [Hewett00] or helps a user build lexicons for the object set*. Context expressions are common strings that are always adjacent to highlighted data on sample pages. Keywords are common words or synonyms that appear near, but not always adjacent to, the data (e.g. common words within a sentence). The BYU ontology-based extraction engine makes use of context expressions and keywords as a data filter when a data item in a single record matches the extraction pattern for multiple object sets. In this work, our ontology generator also makes use of context expressions and keywords as an extraction-pattern filter when more than one extraction pattern in the date-frame library matches with desired values. When a user provides our system with sample data by filling in forms, our system considers words or strings near the highlighted data as possible context expressions or keywords. Specifically, it considers words or strings from the same text node or adjacent text nodes in HTML tree representations of sample records. Then, our system identifies common context expressions and keywords by using string comparisons. The algorithms or heuristics of identifying context expressions and keywords is part of this work. The system may include additional keywords not in the text by using a synonym dictionary or thesaurus. Once the matching is done, and no more results are obtained, then the same algorithm is used for word-by-word matching. Each word is ranked based on its occurrence and finally results are given back to the user. The result evaluation of the proposed method is given in the next section.

5. RESULTS AND ANALYSIS

We compared the results of the proposed algorithm with and without big-data analysis. Wherein delay and correctness of response was observed and final evaluations were performed. In order to evaluate the correctness of the chat-bot response, we formulated the parameter "correctness score" for the chatbot. Correctness Score is the ratio of the total number of correct results produced by the chatbot to the number of results produced in total. It can also be termed as the accuracy of the chatbot. This accuracy was evaluated against different input combinations. The Reuters datasets were used in order to evaluate both correctness scores and the delay of processing. Delay was evaluated based on the total time needed for the chatbot to produce a relevant response after uploading the query. This delay is inclusive of the delay of presenting the results and the delay needed in processing the query on the chatbot's server, which is based on a PHP-MySQL implementation. The following results showcase the correctness score of the chatbot,

Table 1: Correctness scores of the algorithms

Input Query (chars)	CS Pragmatic	CS NLP	CS Proposed
10	0.80	0.85	0.9
20	0.82	0.83	0.91
50	0.76	0.79	0.92
100	0.79	0.80	0.92
200	0.81	0.81	0.93
500	0.82	0.82	0.93
1000	0.83	0.84	0.93
2000	0.83	0.85	0.94
5000	0.84	0.85	0.94
10000	0.84	0.85	0.95

From the correctness scores it is clear that the proposed algorithm produces better results than the existing pragmatic approach and natural language processing [15] approach. This improvement is due to comparison of hash of each and every input word with the given corpus, and therefore requires a lower delay than the existing method. The delay comparison is shown in the following table,

Table 2: Results for delay

Input Query (chars)	Delay (ms) Pragmatic	Delay (ms) NLP	Delay (ms) Proposed
10	1.20	1.62	0.80
20	2.60	2.91	1.50
50	5.90	7.27	5.10
100	10.80	12.18	9.72
200	22.60	25.63	20.80
500	56.71	59.68	49.89
1000	113.35	122.59	100.02
2000	226.33	260.36	200.57
5000	565.51	590.66	501.32
10000	1131.60	1179.28	1003.00

Thus, by improving on the delay of the system, the overall speed of the chatbot can be improved. Moreover, the proposed chatbot performs better than pragmatic and NLP-based approaches, which is a big leap in terms of chatbot performance.

6. CONCLUSION AND FUTURE WORK

Due to an increase in the correctness score of the chatbot by more than 12%, the proposed chatbot is more effective for real-time applications. Moreover, the overall chat bot structure is made so flexible that it can support any kind of text-based documents for producing correct results. The proposed work's delay is reduced, due to the extensive hash-based scanning needed for a highly accurate system. Due to the emerging nature of block chain based techniques, researchers can try to integrated block chain into the proposed chatbot and observe the result improvements in chatbot's authentication performance and removal along with the overall security of the bot. This will help the researchers to further study the effects of block chain in chatbots and explore more areas in the field of application.

REFERENCES

1. Peng, Z., Ma, X. A survey on construction and enhancement methods in service chatbots design. *CCF Trans. Pervasive Comp. Interact.* **1,** 204–223 (2019). https://doi.org/10.1007/s42486-019-00012-3
2. G. Daniel, J. Cabot, L. Deruelle and M. Derras, "Xatkit: A Multimodal Low-Code Chatbot Development Framework," in *IEEE Access*, vol. 8, pp. 15332-15346, 2020.
3. Liu, Q., Huang, J., Wu, L. *et al*. CBET: design and evaluation of a domain-specific chatbot for mobile learning. *Univ Access Inf Soc* (2019).
4. Griol, D., Callejas, Z.: An architecture to develop multimodal educative applications with chatbots. Int. J. Adv. Rob. Syst.
5. Benotti, L., Martínez, M.C., Schapachnik, F.: Engaging high school students using chatbots. In: Proceedings of the 2014 Conference on Innovation and Technology in Computer Science Education, pp. 63–68. ACM, New York, NY, USA (2014) https://doi.org/10.1145/2591708.2591728
6. Jia, J.: CSIEC: a computer assisted English learning chatbot based on textual knowledge and reasoning. Knowl. Based Syst. **22**(4), 249–255 (2009).
7. Coniam, D.: The linguistic accuracy of chatbots: usability from an ESL perspective. Text Talk. **34**(5), 545–567 (2014)
8. Kim, N.Y.: A study on different types of speech acts in voice-chat between EFL students and a chatbot. Stud. Engl. Educ. **22**(3), 81–109 (2017) https://doi.org/10.22275/SEE.22.3.04
9. Shawar, B. A., Atwell, E.: Different measurements metrics to evaluate a chatbot system. In: Proceedings of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies, pp. 89–96. Association for Computational Linguistics (2007)
10. Yu, Z., Xu, Z., Black, A.W., Rudnicky, A.: Chatbot evaluation and database expansion via crowdsourcing. In: Proceedings of the chatbot workshop of LREC, pp. 102–107. Association for Computational Linguistics (2016)
11. Radziwill, N.M., Benton, M.C.: Evaluating quality of chatbots and intelligent conversational agents. CoRR abs/1704.04579 (2017)
12. Wu, Y., Li, Z., Wu, W., Zhou, M.: Response selection with topic clues for retrieval-based chatbots. Neurocomputing **316**, 251–261 (2018). https://doi.org/10.1016/j.neucom.2018.07.073
13. Hwang, S., Kim, B., & Lee, K. (2019). A Data-Driven Design Framework for Customer Service Chatbot. In A. Marcus, & W. Wang (Eds.), *Design, User Experience, and Usability. Design Philosophy and Theory - 8th International Conference, DUXU 2019, Held as Part of the 21st HCI International Conference, HCII 2019, Proceedings* (pp. 222-236). (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Vol. 11583 LNCS). Springer Verlag
14. Chowanda, P. Blanchfield, M. Flintham, M. Valstar Computational models of emotion, personality, and social relationships for interactions in games. In: The 2016 International Conference on Autonomous Agents & Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems (2016), pp. 1343-1344
15. Galitsky, B., Ilvovsky, D., and Makhalova, T. (2019). Developing Enterprise Chatbots, Springer Switzerland https://doi.org/10.1007/978-3-030-04299-8
16. Joshi, Sujata: Applications of Chatbots in Marketing: Use Cases, Impacts, Challenges and Drivers Volume 8, No.1.6, 2019 ISSN 2278-3091, International Journal of Advanced Trends in Computer Science and Engineering, 8(1.6), 2019 https://doi.org/10.30534/ijatcse/2019/3081.62019
17. Nishad Nawaz: Artificial Intelligence interchange human intervention in the recruitment process in Indian Software Industry ISSN 2278-3091 International Journal of Advanced Trends in Computer Science and Engineering Volume 8, No.4,July-August 2019.