Journal of Theoretical and Applied Information Technology
15th February 2022. Vol.100. No 3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AN AGENT-BASED DOCUMENT CLASSIFICATION MODEL TO IMPROVE THE EFFICIENCY OF THE AUTOMATED SYSTEMATIC REVIEW PROCESS

MOUAYAD KHASHFEH[1], MOAMIN A MAHMOUD[2], MOHAMMED NAJAH MAHDI[3]

[1]College of Graduate Studies, Universiti Tenaga Nasional

[2,3]The Institute of Informatics and Computing in Energy (IICE), Universiti Tenaga Nasional, Malaysia

[1]mou2ayad@gmail.com, [2]moamin@uniten.edu.my, [3]najah.mahdi@uniten.edu.my

ABSTRACT

This paper proposes an Agent-based Document Classification (AbDC) model that computerizes the systematic literature review (SLR) process by imitating what a researcher is supposed to perform during the literature review process manually. The AbDC model comprises three main components that perform the SLR. Firstly, the document classification algorithm analyses a full text of research articles and evaluates relevancy. Secondly, the multi-agent architecture accelerates the mining process and handles the performance issues. Finally, the web-based systematic review tool tests and validates the functionality of the proposed AbDC model. The first testing was conducted to assess the performance of the proposed AbDC. Result shows that the required processing time was reduced by 33.5% using four agents to achieve the mining process. Meanwhile, the second testing was performed to validate the mining process results. The text extraction method was run on 200 documents from various studies to conduct the review process. The parsing process yielded valid results with 98.5% accuracy. The testing results showed that the proposed AbDC model is significant in providing researchers and postgraduate students with new means to perform SLR.

Keywords: *Document Classification, Systematic Literature Review, Multi-Agent System.*

1. INTRODUCTION

The process of performing literature review consumes a considerable amount of time and effort from researchers and students. Such a process requires reading and understanding hundreds of studies and subsequently extracting and excluding relevant or irrelevant articles, respectively. In a remarkably advanced method, the current trend is the application of the systematic literature review (SLR) concept, which involves a sophisticated procedure to ensure valid, comprehensive and highly relevant outcomes. By contrast, finding information on the WWW was markedly enhanced when using search engines in the mid-1990s. However, the development of science and the spread of knowledge coincide with a growing number of publications, and the volume of online content continues to grow rapidly. The search engines for some submitted queries may retrieve documents of low relevancy to the queries. Additionally, the number of digital materials is tremendously increased. In May 2011, Google indexed an estimated 35 billion web pages, and the number has risen to 45 billion after two years. A large number of returned results by search engines for a submitted query requires reading and discovering the most relevant, thereby discarding more than half of what has been read. 'Systematic review' has become one of the top trends in the research community and has dramatically increased in the last few years.

A primary objective of any systematic review is to minimise publication bias [1] by analysing all studies relevant to the review. Scells 2018 [2] identified systematic review (SR) as a type of literature review that appraises and synthesises the work of primary research studies [3] to answer one or more research questions. SLR is a rigorous approach for stand-alone literature review; an SLR is when a stand-alone literature review is conducted using a systematic, rigorous standard [4]. Fink [5] introduced an SLR as a systematic, explicit, comprehensive and reproducible method for identifying, evaluating and synthesising the existing body of completed and recorded work produced by researchers, scholars and practitioners. The systematic review [6] focuses on a specific question to provide evidence to underpin a part of the research. The stand-alone part of the study

should be performed before conducting further research [7]. This part requires searching through several particularised data sources using specific search terms. Table 1 summarises the systematic review challenges.

Table 1. Systematic review challenges

Technical knowledge	The researchers might be unaware of systematic review methodology (i.e., performing a narrative review) [21].
	Systematic review uses some advanced tools and applications to organise the data, and this kind of application is either expensive or difficult to use for non-technical users.
	Most of the available tools focus on organising the metadata [22] of the studies but do not offer advanced features to help researchers in inclusion/exclusion tasks. Thus, screening and evaluating processes on behalf of the researcher and providing reliable suggestions with an acceptable level of accuracy are necessary [19].
	Using the available applications and tools requires a level of technical experience from the researcher to efficiently utilise these tools, which is not always available in non-technical researchers, such as the researchers in (human rights, geography, history, and human resources) [21].
Time consumption	Conducting a systematic review is time consuming. The user must collect and organise the metadata of each study used in the research to generate statistical reports and charts from the data aggregation [22].
	Screening the text of the study for inclusion and exclusion is a time-consuming task [22]. This task will take a considerably long amount of time and effort from the user regardless of whether the user screens the document fully or partially (Abstract and Title) and wants to follow the two-step screening approach [23].
Search accuracy	Some researchers might not follow the inclusion/exclusion criteria set when starting the research [19]
	Researchers lack a unique method (criteria) for exclusion/inclusion [21], which impacts the research accuracy. For example, suppose 20 studies are given to three researchers to include and exclude relevant and irrelevant studies. Thus, each researcher might return different results based on the criteria and the method they followed. The reliability of every survey is determined by defining whether the study's accuracy is 'high' or 'low'.
	Some researchers might not use a two-step screening process (firstly reviewing Title/Abstract than the entire study) [22]. Two screening processes may need a long time; thus, many researchers conduct only one-step screening (title and abstract screening) [23].

The conduct SR process [14][15][17][18][19][20], a complete collection of keywords is formulated and utilised to query different databases, such as PubMed and Web of Science, to obtain all the relevant studies. Afterwards, the titles and abstracts of the studies should be reviewed individually and verified versus a kit established previously of acceptance criteria for inclusion and exclusion. The study should also pass the first elimination process. The entire text of the study is obtained and tested to decide its involvement or elimination from the review. An SR attempts to collate all experimental proof that meets pre-specified qualification rules to find the answer to a particular research inquiry. SR also utilises explicit, systematic techniques chosen to reduce bias, producing reliable conclusions to obtain outcomes and present judgements. Figure 1 shows the SR steps for conducting a systematic review [16].

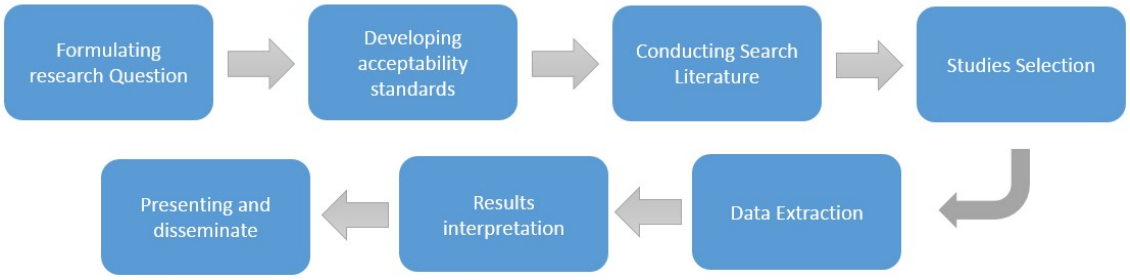


Figure 1. Steps to conduct an SR

Many research studies attempted to use technologies to facilitate the systematic review process [8][9][10]. Text mining [11], which is one of these technologies, is used to conduct systematic reviews to reduce the screening workload by automatically eliminating irrelevant studies. Using text mining in the SR process, which is not yet fully proven, is considered promising. Although this method may be used with a high degree of confidence in many fields, additional developmental and evaluative work is needed in other disciplines. Using text mining to perform systematic reviews can save effort and time. However, a serious concern related to the efficiency of the process considering the performance, where text mining is regarded as a heavy process and costly considering processing and time [8]. One of the proposed solutions to tackle the performance issue of text mining is using the multi-agent system (MAS) [33][34][35][36][37][38]. However, some research studies showed the significance of finding a way to overwhelm the performance issue when applying text mining algorithms [12][31][32][40].

A MAS provides the advantages of computational efficiency, reliability, extensibility, maintainability, robustness, maintainability and responsiveness, flexibility and reusability [24]. Interaction between agents can be conducted through an agent communication language. The MAS has been applied for various applications [39] [40][41][42], including natural language processing [29] and data mining [25]. The MAS theory can typically be adapted for two main reasons [28]. Firstly, data are often intrinsically distributed over several sites. Collecting these data in a single processing node is inconvenient for the communication costs, feasible for privacy policies and even beneficial for real-time data streams. Secondly, the complexity of the task is often. Thus, the data must be partitioned to process each data subset independently and finally combine the partial results. Multi-agent systems [30] is a subfield of distributed artificial intelligence, which has experienced rapid growth because of its flexibility and intelligence to solve distributed problems [43][44][45][46][47]. According to [26], well-documented advantages of the MAS include decentralised control, robustness, simple extendibility, sharing of expertise and sharing of resources. Consequently, some of the benefits available when using multi-agent technology in large systems are presented as follows.

- An increase in the speed and efficiency of the operation due to parallel computation and asynchronous operation [30]
- Remarkable degradation of the system when one or more of the agents fail, thus increasing the reliability and robustness of the system [30]
- Scalability and flexibility: Agents can be added as and when necessary [27]:
- Reduced cost: Individual agents cost remarkably less than a centralised architecture [27]
- Reusability: Agents have a modular structure and can be easily replaced in other systems or upgraded more quickly than a monolithic system [30]

Several tools and systems have been developed to help perform the systematic review process by providing some features to assist researchers in one or multiple stages of the systematic review. However, most of these tools and systems, such as EndNote and Rayyan, focus on the meta-analysing part and organising the references. By contrast, only a few tools and systems focused on automating the screening stage to aid in taking inclusion/exclusion decisions versus a predefined punch of inclusion criteria; for example, but not limited to the EPPI-Reviewer system [13], which provides features regarding metadata services (collecting, analysing, organising and reporting), as well as full-text screening service by using their text-mining algorithm. However, this service is unavailable for all users. They have a real performance problem due to the heaviness of the text mining process, considering the time needed to analyse the studies and the processing because it excessively consumes

the CPU memories. Thus, if many users are allowed to leverage this service, then the analysis results may not be provided within a reasonable time. Moreover, users employ massive resources to parse a vast volume of studies. The EPPI-Reviewer system has remarkable features. However, using this system is difficult, especially for those who do not have technical skills and are unfamiliar with using the software effectively (such as the researchers in Humanities, Law, Chemistry, History and other non-technical domains). Finally, the text mining algorithm of the EPPI-Reviewer system [13] can be improved to give accurate results by considering additional factors for enhancing text mining, which mainly focuses on the terms of frequency factor to extract the keywords from the text. This paper proposes an AbDC model that executes a document classification algorithm through multiple agents simultaneously to analyse many documents to overcome the slowness issues caused by the heaviness of the classification process. The current study is directed to scientific publications with short study documents (less than 25 pages per document) and available in pdf format as English text (not images or handwritten).

2. AGENT-BASED DOCUMENT CLASSIFICATION (ABDC) MODEL

This section presents a combination of text mining in terms of document classification and data mining with a multi-agent system to extract the potential context of scientific publication from a particular scientific publication warehouse and thus define which publication is potentially relevant to a searcher's need. The designed algorithm gives the user the ability to input the keywords besides assigning a threshold value for each one. The algorithm consists of three processes, detection process, preparation process, and mining process. The detection process includes the determination of document language and abstract, and keywords. The Preparation consists of the processes, split content to paragraphs, paragraph length determination; converting text to lower case; text typography factor, content tokenization, removing stop words. Finally, the mining includes the processes, regular expression; normalization; grouping, and computing frequency. The designed algorithm would be helpful in providing an alternative means of searching highly relevant content from large databases.

The proposed model consists of five components: the interface, parsing process, storing process, notifying process, and Search process, as shown in Figure 2. The interface provides communication means between the user and the agents in the backend process. Besides, it provides an input tool for users' search preferences. The second component is the parsing process that operates by a document classification algorithm. The third part is the storage process that manages by DB Agent. The fourth component is the notifying process executed by the personal agent and the monitor agent. The last component is the search process that is operated by pattern matching, where the user submits the search query to the engine through the interface to run the search process.

2.1. The Parsing Process

The processing flow of the AbDC model starts (1) by creating an independent repository for research by the user and feeding the system with the desired keywords from the user based on the topic. Figure 3 describes the parsing process flow. The user should collect documents related to the topic and add them to specific storage in PDF format, which can be cloud storage or local storage. The user can also upload the documents as PDF through the interface, Web UI. (2) All uploaded documents flow to a queue and wait to be analyzed. Thus, the user will not get the result of analyzing his document immediately (not a real-time process) since the platform serves multiple users with variant keywords and topics. The Mining Agents Coach (MAC) plays organizer role of the parsing queue (PQ) and the mining agents, MAC dequeues the documents from the queue (3) and gives each document to agent individually on parallel to parse it, Parsing document can be achieved by applying the document classification algorithm on each document to extract the keywords and the terms along with the thresholds relevancy for each keyword in the document, MAC has the power to create and drop mining agents (MAs) based on the number of the documents in the queue, which improves the performance of the process by handling many documents at the same time, MAC should guarantee the Agents Coordination, means each one agent should work on one document at the same time, in order to avoid working on one document by multiple MA agents, and should guarantee the data integrity, means, each MA agent should deliver the output of the mining process on the assigned document before getting a new one, and the document would not be removed from the queue unless MAC gets a feedback from the MA on charge to process that document, hence, MAC can pass a document from agent to another agent in case of failure one of the MA agents.

Journal of Theoretical and Applied Information Technology
15th February 2022. Vol.100. No 3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

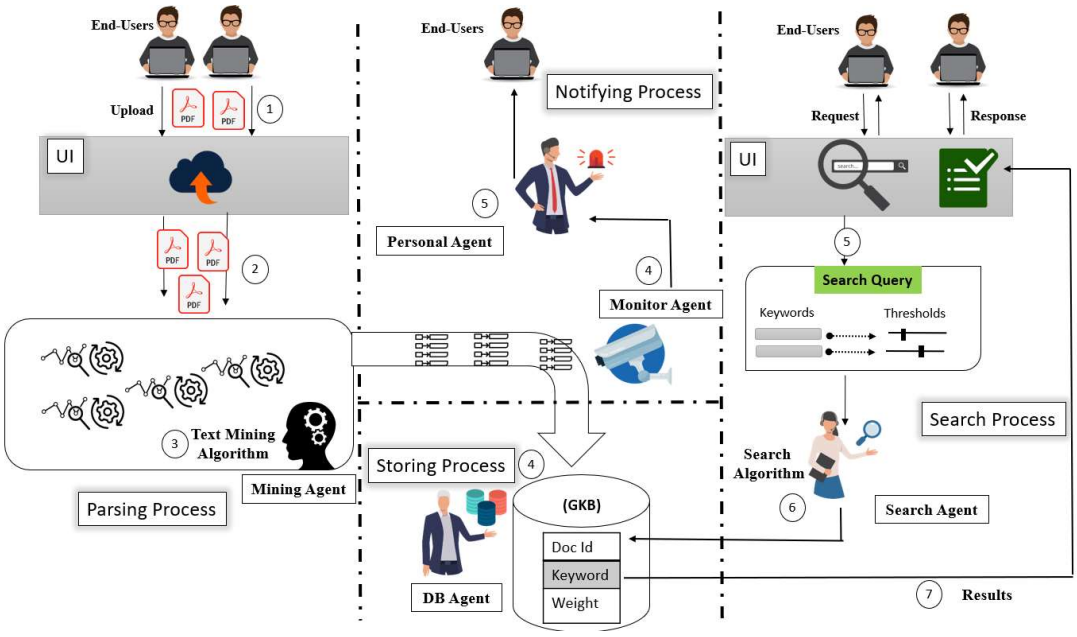


Figure 2. Agent-based Document Classification model

After the MA agent finishes parsing a document by extracting the keywords and the thresholds (4), it sends these results to the DB agent (5) and gives a heads-up to the MAC that the task is done, then MAC might assign a new document to that agent or drop it.

2.2. The Storing Process

Saving in a structured database is the final step after analyzing the document's text, extracting keywords, and computing. The threshold for each keyword is storing by considering all the previous factors. These keywords and the threshold value in the queryable database are considered so the user can search through this database for the keywords without analyzing the document again. The document after the analysis would be available for the user to search and use. Storing the data in this way allows us to collect the rows individually or in aggregated shape. Hence, the user can obtain the data appropriately by organizing the data within a table or spreadsheet. The most benefit for storing the data in the relational database is requesting the documents containing one or many keywords in a specific threshold, ordering them by relevancy, and reverting the data to the user across a friendly graphical user interface. The aggregated format is very useful to present the data through charts and graphic lines. As shown in Figure 4, the DB agent collects the results (1) (keywords and thresholds) and the document id, and the repository id that the document belongs to from the MA agents. Then it matches (2) the extracted terms with the desired keywords added by the user upon creating the repository of the research by removing the non-required term from the results if the term is not matching any of desired keywords. The final step is storing (3) the needed terms along with the weight (threshold) in the relational database that we called Global knowledge Base (GKB).

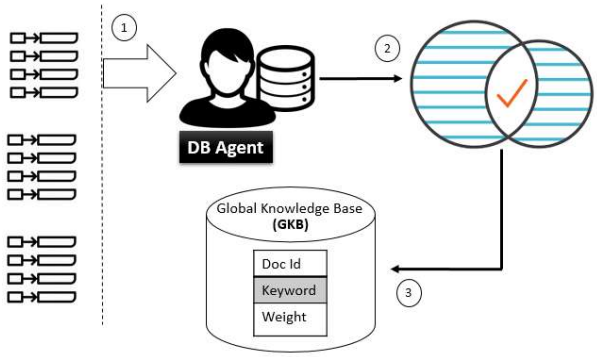


Figure 4. The storing process

Journal of Theoretical and Applied Information Technology
15th February 2022. Vol.100. No 3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

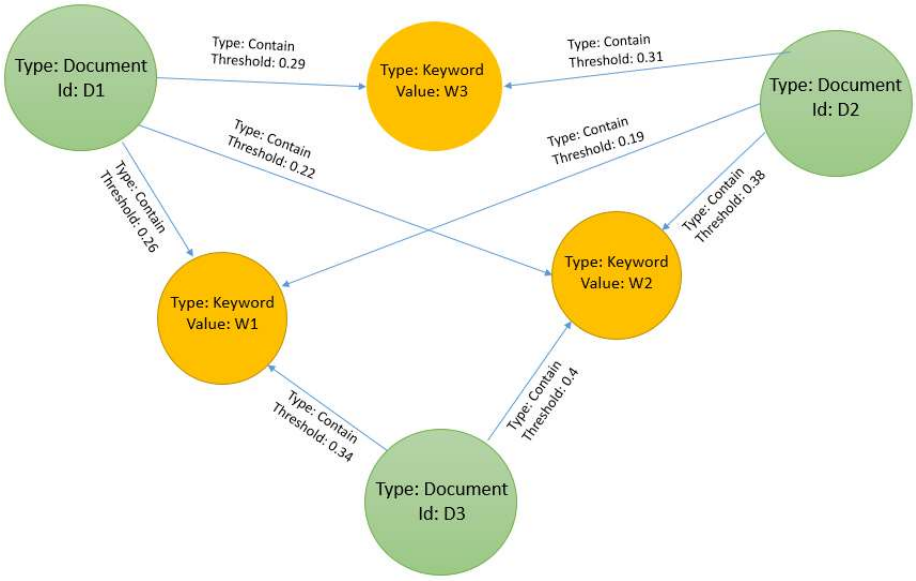


Figure 5. Graph Database Document-Keyword

The keywords and the documents can be saved in a Semi-structured database (Graph DB) as well, in the form of nodes, relations, and attributes. By saving the data, we can build a recommendation system by applying some machine learning algorithm to recommend other documents that can be similar to a document that the user found useful and relevant to the desired topic. Hence, the user can search for documents covering a specific topic by using documents instead of keywords. Therefore, if the user finds a document using few keywords and marks it as useful, the system can recommend other documents that can be similar to that document. Figure 5 shows an example of saving the data within Graph Database. It consists of many nodes with different types (Document or Keyword) and Properties (Value, Threshold, Id), and relationships (Contain).

2.3. The Search Process

The main aim of analyzing documents and extracting the keywords, and saving them inside the database is to make this data is quarriable. So the user can send his desired keywords along with the assigned thresholds to obtain the documents meeting the sent query and ordering the results by the similarity against the search query from highest to lowest. This section explains the process and the search algorithm to get the documents based on the similarity with a given query. Figure 6 shows the search process. It (1) starts by inserting desired keywords along with thresholds relevancy for each keyword. Then, the search agent (2) collects the input data and (3) starts searching inside the GKB database to (4) find the documents containing the desired keywords with threshold greater than or equal assigned thresholds by applying the designed search algorithm. Finally, the agent (5) returns the results to the User interface with the possibility of saving the search for this user to send him/her further results later once a new document is analyzed and meets his search conditions. In this case, the system creates an agent on behalf of the user to provide him with these extra results; the generated agent would stay alive for a specific period specified by the user.

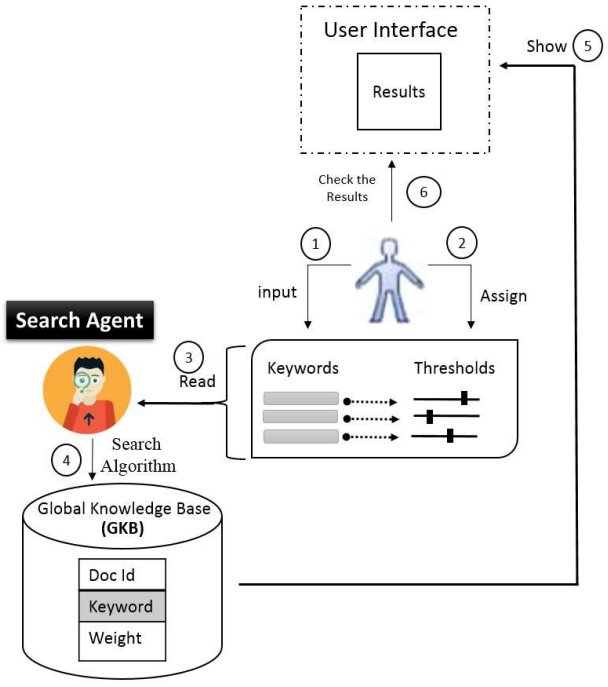


Figure 6. Search Process

To build the search algorithm, we rely on the Vector-Space algorithm and modify the top of the algorithm to fit our case (long documents) since the large articles are poorly represented using Vector-Space because they have poor similarity rates. Let's assume that the search engine received the search query from the user, consisting of multiple keywords and thresholds value for each keyword

$$q = \left(qkw_0^{thr0}, qkw_1^{thr1}, qkw_2^{thr2}, qkw_3^{thr3}, \dots\dots\dots, qkw_{n-1}^{thr(n)}\right) \quad (1)$$

Where q is the query, qkw is a keyword within the query, thr is the keyword threshold, n is the number of keywords in the query.

The Searching process steps to return the results for this query are as the following:

- Counting the total number thresholds Sum_q^{thrs} of all keywords in the query q:

$$Sum_q^{thrs} = \sum_{i=0}^{n-1} thr_i \quad (2)$$

$$\downarrow$$

$$TotalThresholds_q = (thr_0 + thr_1 + thr_2 + thr_3 + \dots\dots\dots + thr_{n-1}) \quad (3)$$

- Retrieving TF(term of frequency) for each qkw for each document d from the relational database containing the analyzed documents ($\boldsymbol{tf_{qkw}^d}$)

$$\boldsymbol{tf_{qkw}^d} = the\ \boldsymbol{tf}\ of\ the\ keyword\ \boldsymbol{qkw}\ within\ the\ document\ \boldsymbol{d} \quad (4)$$

- Tokenize the keywords and apply the stemming algorithm used in the parsing document process to extract the stemmed keywords from the query q.
- Calculating the relevancy REL of each document d against the query q (REL_d^q) , which equals the sum of the relevancy of each query keyword qwk in the query (REL_d^{qkw}):

$$REL_d^{qkw} = tf_{qkw}^d \times \left(\frac{thr}{Sum_q^{thrs}}\right) \quad (5)$$

$$hence \downarrow$$

$$REL_d^q = \sum_{i=0}^{n-1}\left(tf_{qkw_i}^d \times \left(\frac{thr_i}{Sum_q^{thrs}}\right)\right) = \sum_{i=0}^{n-1}\left(REL_d^{qkw_i}\right) \quad (6)$$

- The last step is ordering the documents by the relevance value (REL_d^q) from the most descending

2.4. The Notifying Process

As shown in Figure 7, when a user searches for documents using particular keywords with specific thresholds relevancy, he/she can enable the option "Notify me" as well. This option means that the user can get more input parameters from the future documents uploaded by the user, which would be added to the storage or analyzed later. Hence, the system assigns the user's requests/inputs to the personal agent in order to start finding relevant documents on behalf of the user and provide the user with new results from the future documents accordingly. The user should also specify the period of the monitoring, for example, for ten months. In this case, the Monitor Agent MOA for this request would still be active for ten months and keep providing the user with further results for this period.

After inserting the inputs (keywords, thresholds relevancy, Monitoring Date), the MOA reads these inputs and search inside GKB and return the results to the user. This agent would save the search request inputs inside another database called Monitoring database containing request inputs as well as requester email and monitoring validity period. As shown in Figure 7, the parsing process mines documents, extracts (7) keywords and their thresholds relevancy, and store these results inside GKB. During submitting the results to GKB, MOA (6) reads the users requests from the monitoring database and (8) watches parsing process results. If any results meet any user request, the PA would save these results and (9) notify the requester later by email or a notification on the website that he/she has further results to his request.

3. THE DOCUMENT CLASSIFICATION ALGORITHM

This section presents the designed algorithm to identify relevant studies from large databases that include thousands of articles. The algorithm architecture consists of eleven stages, as shown in Table 2 and Figure 8:

Journal of Theoretical and Applied Information Technology
15th February 2022. Vol.100. No 3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

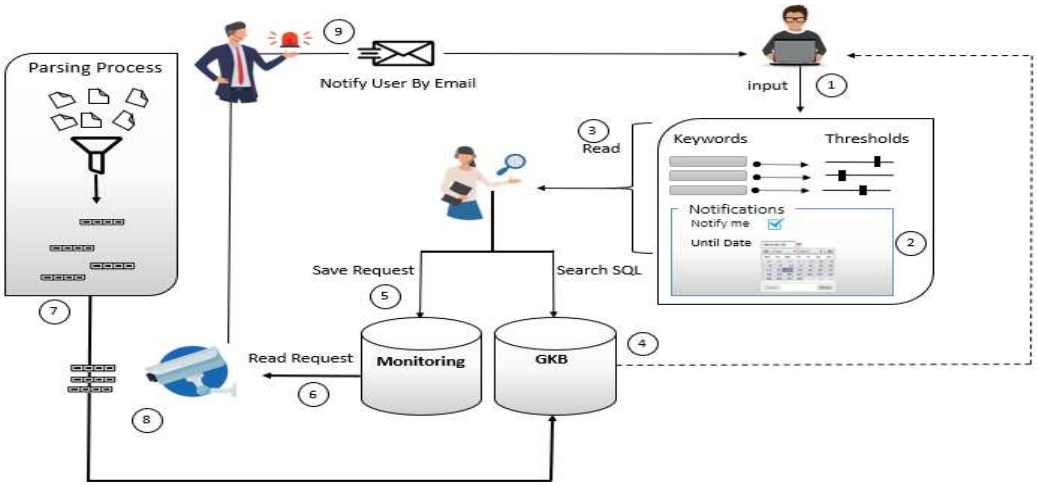


Figure 7 Personal Agent (PA) and Monitor Agent (MOA)

Table 2 Document Classification Algorithm Stages

#	Stage	Summary	Technique Applied
1	Words and Fonts Extraction	To extract the words and fonts from PDF documents	iTextSharp library
2	Language Identification	To determine the document language to exclude non-English	Google Translator API
3	Identifying the abstract and keywords sections	To Identify the words that appear in the abstract and keywords sections to give them higher weigh	Open NLP
4	Content Tokenization	To split the text into many pieces such as words, phrases, symbols, keywords	OpenNLP
5	Converting text to lower case	To convert all the words to the small letters only, to facilitate the comparison and gathering	.Net Text library
6	Text transformation (attributes generation)	Creating entity or bag to each word holding information about the word (order, length, stem value ,stop word, font)	Class (object-oriented programming)
7	First Filter - Removing stop words	To eliminate the words that attributed as stop words from the words collection	.Net Linq library
8	Second Filter- Regular expression	To eliminate the words that don't meet a regular expression, we chose (length more than 3 letters and alphabets only)	.Net Regular expression library
9	Normalization	To generate the stemmed value from the word	Porter Stem algorithm
10	Replacing keywords Options	To Replace synonyms words with original words	.Net Linq library
11	Synthesizing based on the attributes	To group the words by their attributes (order, Stem, Font) into lists and maps	.Net Linq library
12	Document Text Typography Factor	To compute the value of TTF for each document and identify TTF benchmark	TTF formulas
13	Calculating keywords weight	To calculate the weight of each keyword in the document based on (TF and TTF) factors	Term Frequency
14	Extracting expressions	To extract expressions from the document, the expression contains 2 or 3 words	.Net Collection library
15	Storing outputs in a relational database	To store the extracted expressions, keywords, fonts and weights into a relational database	SQL Server DB

Journal of Theoretical and Applied Information Technology
15th February 2022. Vol.100. No 3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

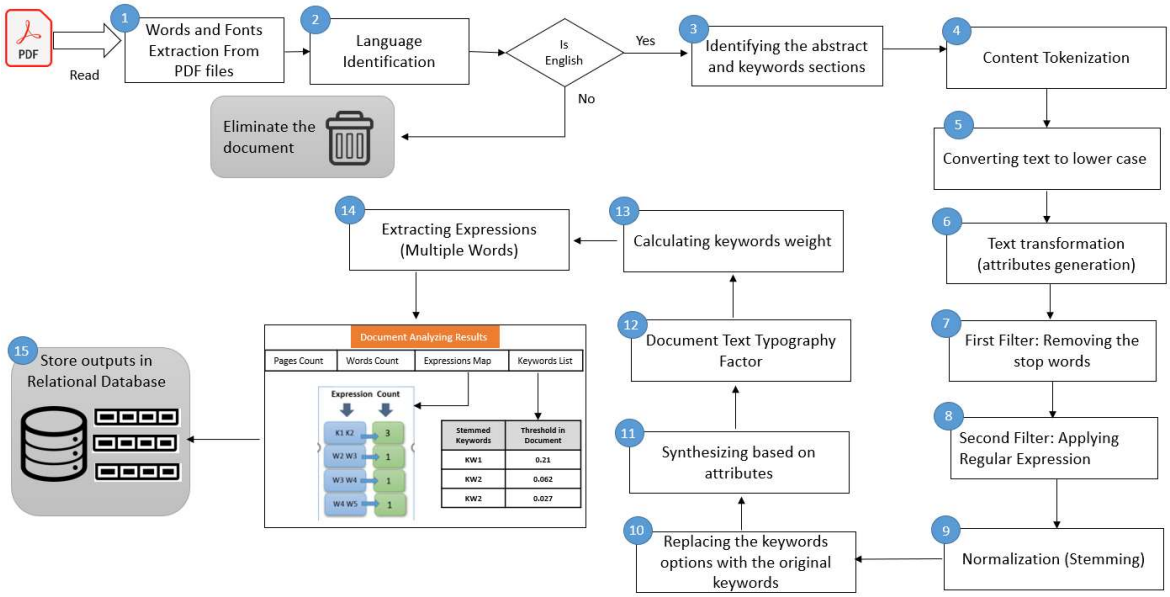


Figure 8 Document Classification Algorithm Stages

Following, we elaborate on the major stages.

3.1. Synthesizing based on Attributes

In this phase, the list of word entities is created and ready to be pushed to the document entity. As mentioned before, there is a method in the document entity that offers this ability. The method of adding word entity to document entity goes through several stages:

- Increase the count of WordsOfDocumecumt by 1.
- Check the validity of the Word entity by calling IsValid() method, which returns "true" if the word isn't a stop word and matches the Regular Expression (Regex), otherwise, the word will be invalid, and the method will returns "false".
- Add a reference to the Word entity to the WordsList collection within the document entity. If the word is valid, otherwise, it would be ignored, the words would be ordered by the position of the word in the document in the list.

Add the word entity to the TermsMap dictionary within the Document entity. If a word is valid, otherwise, it would be ignored. Before adding a word entity to this dictionary (Map), the method checks if the map contains the word's stemmed value in the keys. If yes, then the key would not be added because it already exists on the map. Still, a new Word entity would be added to the list of Word entities and linked to the key containing that stemmed word. Still, if the map does not contain the key, then a new key would be added to the map with the stemmed word, and a new Word entity would be added to the value and linked to the new key, as shown in Figure 9.

- Add the FontSize to the FontCount map. The map's key is the font size, and the value is the count of the words following this font in the document. Therefore, when a new word entity is added, the algorithm will check if this word's font size exists in the FontMap. If it exists, then the count of that font size would be increased by one. In case it doesn't exist, the font size would be added to the FontSize map in the new key, and the count would be 1 in the value, as shown in Figure 10. After adding parsing all the words in documents and creating a word entity from each of them, and adding them to the document entity, the document entity is ready to call the AnalayzeDocument method to start doing the Factors thresholds.

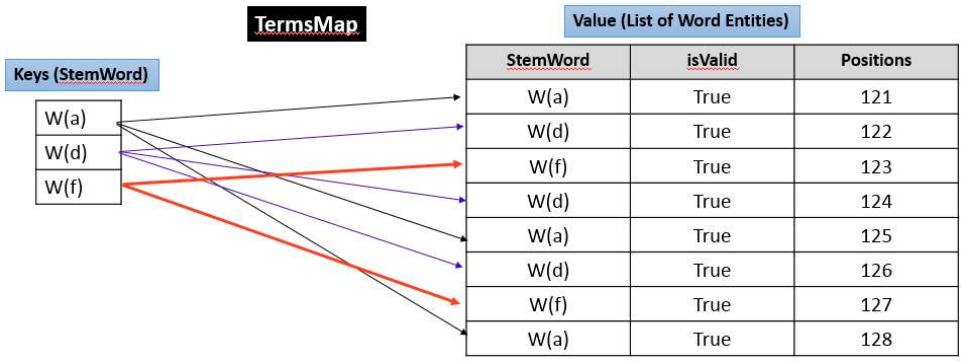
3.2 Text Typography Factor

Text Typography Factor (TTF) of a document is considered to give the higher score, the words having bigger font sizes or in bold case, as shown in Figure 11.

Figure 12 presents a document consisting of several font sizes (12, 18, 24, 40). Some of the words are bold, so to determine TTF, a benchmark should be identified first to compare each font in the document. Therefore, we calculate FontBenchmark as following:

Journal of Theoretical and Applied Information Technology
15th February 2022. Vol.100. No 3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195



StemWord	isValid	Positions
W(a)	True	121
W(d)	True	122
W(f)	True	123
W(d)	True	124
W(a)	True	125
W(d)	True	126
W(f)	True	127
W(a)	True	128

Figure 9. TermsMap

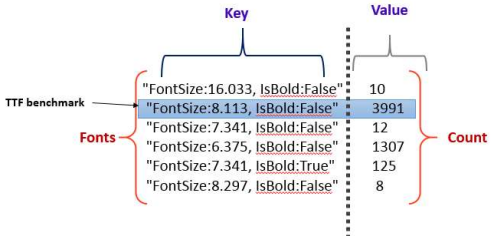


Figure 10. FontCount map

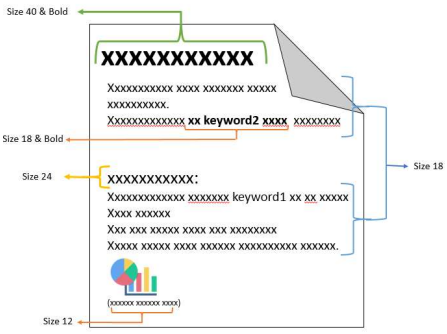


Figure 11. Font features

BiggestCount = Document.FontCount.Max(Font => Font.Count)

FontBenchmark= Document.FontCount.Where(Font.Count == BiggestCount)

The next step is identifying the value of BoldFactor for each font. BoldFactor is a numeric value that can be only one of two options:

- BoldFactor of the font is 0: if the FontBenchmark is Bold, regardless of the font thickness, or the font is not Bold
- BoldFactor of the font is 0.5: if the FontBenchmark is not Bold and the font is Bold

The next step is calculating the TTF value for each font based on the FontBenchmark and the BoldFactor of the font. TTF calculation is done as the following

FOR each Font in FontCount Map from Document Entity

IF FontBenchmark is not Bold AND Font is Bold

Font.BoldFactor=0.5

Else

Font.BoldFactor=0

END IF

IF Font.FontSize is smaller than or Equal FontBenchmark.FontSize

Font.TTF= 1

Else

Font.TTF= Font.FontSize / FontBenchmark.FontSize

END IF

Font.TTF = Font.TTF + (Font.TTF * Font.BoldFactor)

END LOOP

3.3 Calculating Keywords Weight

This section explains the calculation of the weight of each keyword in a document. However, before starting the weight calculation, we need to calculate some factors affecting the weight. The first factor is Term Frequency (TF); as mentioned in Chapter 2, TF is the number of times a word appears in the document (*D*), divided by the total number of words in that document which is 3645 words as per the previous section:

$$TF_{(t)} = \frac{Number\ of\ times\ term\ (t)\ appears\ in\ a\ document}{Total\ number\ of\ terms\ in\ the\ document} \quad (7)$$

$$\downarrow$$

$$TF_{(t)} = \frac{78}{3645} = 0.0213$$

In the Document entity, we organized all words in one map: TermsMap, so it is easy to know the count of each word in the document using this map. We have a beneficial property holding the count of the words in the document, which is WordsInDocument. The second factor is the Text Typography Factor TTF of each extracted keyword. In the previous section, we show how to calculate the TTF for each font available in the document. So the TTF of a keyword is the average TTF of the fonts that the keyword followed of the document. For example, keyword W appears n=78 times in the document (D) in three various fonts F1, F3, F5, where numbers of impressions per font are n1=2, n2=54, n3=22, respectively:

$$TTF_{(w)} = \frac{((TTF_{(F1)} * n1) + (TTF_{(F3)} * n2) + (TTF_{(F5)} * n3))}{n} \quad (8)$$

$$\downarrow$$

$$TTF_{(w)} = \frac{((1.5 * 2) + (1 * 54) + (1.18 * 22))}{78} = 1.06$$

Now, we can utilize both factors (TF and TTF) to calculate the weight of the keyword. The weight equals TF, multiplied TTF of keyword W in document (D):

$$Weight^{D}_{(w)} = TTF_{(w)} \times TF_{(w)} \quad (9)$$

$$\downarrow$$

$$Weight^{D}_{(w)} = 1.06 \times 0.0213 = 0.022578$$

We introduced a different approach to calculating the weight based on the highest number of the frequency among the keywords in the document instead of the number of the words in the entire document, so the term frequency factor formula would be slightly changed to be like this:

$$TFBMTF^{D}_{(t)} = \frac{Number\ of\ times\ term\ (t)\ appears\ in\ a\ document(D)}{Number\ of\ the\ term\ frequency\ in\ a\ document\ (D)} \quad (10)$$

Where TFBMTF means Terms Frequency Based on Max Term Frequency. Thus, the new formula to calculate the weight of keyword w in document D based on max term frequency (Weigh_BMTF) is:

$$Weigh\_BMTF^{D}_{(w)} = TTF_{(w)} \times TFBMTF_{(w)} \quad (11)$$

The last step is applying the previous formulas on all the extracted keywords from the document. For example, in TermsMap, we have all Word entities sharing the same stemming value grouped under one key: stemming, and the Word entity contains Font entity. Hence, we can run a loop on TermsMap's keys and calculate the weight of each key by applying the formulas as following:

FOR each *StemWord w* ***in*** *TermsMap keys* ***from*** *Document Entity*

TermsList= Document.TermsMap[w]
TTF(w) = TermsList.Average(term => FontsMap[term.Font].TTF)
TF(w)= TermsList.Count / Document.WordsInDocument
Weigh(w) *= TTF(w) * TF(w)*

MaxTermFrequancy = Document.TermsMap.Max(Value.Terms.Count);
TTFBMTF(w)= TermsList.Count / MaxTermFrequancy
Weigh_BMTF(w) *= TTF(w) * TTFBMTF(w)*

END LOOP

The main reason behind having two weights for each term ($Weigh_{(w)}$, $Weigh\_BMTF_{(w)}$) because we use $Weigh\_BMTF_{(w)}$ for searching using the keyword and threshold as input from the user, on the other hand, $Weigh_{(w)}$ It is important to order the retrieved documents by relevancy because the document's length affects the relevancy factor, as mentioned before. Thus, the big document containing a lot of pages is considered less relevant to a topic comparing to a short document containing fewer pages and words, even though, $Weigh\_BMTF_{(w)}$ for some keywords are equal among both documents. For example, the keyword *k* has the biggest weight in document *D1,* which contains n number of words, and document *D2,* which contains *n*5* words. Hence, both documents are very relevant to *k.* However, the document *D1* is more relevant to *k* comparing to document *D2*:

$$Weight^{D1}_{(k)} = Weigh\_BMTF^{D2}_{(k)} = 1 \quad (12)$$

BUT

$$Weight^{D1}_{(k)} > Weigh\_BMTF^{D2}_{(k)} \quad (13)$$

3.4 Extracting Expressions (Multiple Words)

The designed algorithm offers another feature that might be useful to the clients, which is extracting expressions from the document. The expression contains 2 or 3 keywords.

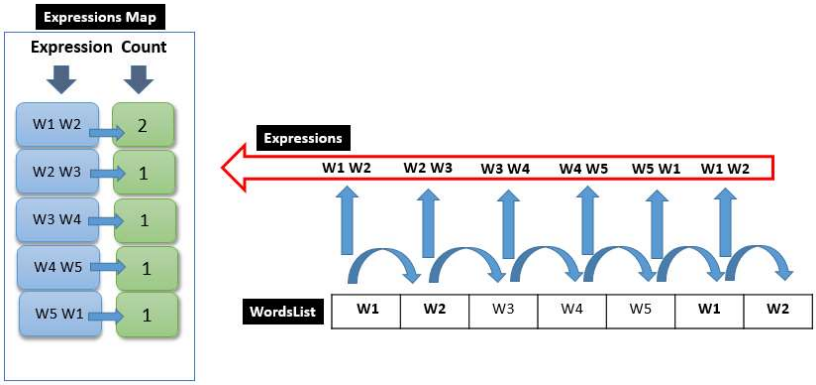


Figure 12. Extracting Expressions process

As shown in Figure 13, we consume the words existing in the WordsList collection in the Document entity. WordsList contains all the words of the document in the stemmed case, ordered by the advent of the words in the main document. The algorithm creates an expression from every 2 and 3 consecutive words and adds the expression to the Expressions Map, where the Expressions Map is a (key, value) Map, it contains the expressions as keys and the appearance frequency of the expressions as a value, thus, after combining two or three consecutive words to create an expression, we check if this expression exists in the map, so if the expression is already added before, then we don't add it again since the key in the map is unique, but we will increase the appearance frequency of that expression by 1. Otherwise, we add a new key containing the expression, and the appearance frequency equals one as the initial value. We mentioned the algorithm could be fed with a list of keywords and options or synonyms of the keywords to help the algorithm give more accurate results. In case a document contains one of the synonyms instead of the original keywords, the algorithm accepts the expressions as well, so the algorithm would replace the synonyms or the expression with the original expression during the expressions extraction process. This feature makes the algorithm more flexible and improves its accuracy.

For example, the multi-agent system concept has another scientific expression which is Distributed systems, so the user can teach the algorithm to replace the expression "Distributed systems" with the "multi-agents system" expression if the first expression is found. Thus, the user doesn't need to search for "multi-agent system" and "distributed systems" independently, where one of them can be set as an option to another before starting the parsing process. Later, users can search just by using the main expression. The algorithm stores the expressions and the expression's options in the memory before starting the extraction process from the documents in the stemmed format. The algorithm checks if any extracted expression matches one of the expression's options provided by the user, so it reverts the expression to the original expression in stemmed format, as shown in Figure 13.

Figure 13 shown an example of matching and replacing expression. The expressions options map contains the expressions (W1 W2) and (W5 W1) as options to the main expression (K1 K2). Before adding the expressions to the extracted expressions map, we check if the extracted expression (W1 W2) is listed as an option to another expression, so we replace (W1 W2) and (W5 W1) with (K1 K2) and add (K1 K2) to the final map.

4. IMPLEMENTATION OF THE SYSTEMATIC LITERATURE REVIEW MODULES

In this section, we introduce the entire process we develop to mitigate the systematic review process, starting from the Web facade that used by the users to upload the documents to the system, feeding the system with the main keywords, inserting metadata for each document manually and using Google Scholar API, parsing the documents files and extract the keywords along with thresholds, providing a friendly user interface to find a document through the metadata, generating aggregation reports and charts based on the metadata fields, providing direct links to related works and versions of a document on Google Scholar, collecting the citation text from Google Scholar if applicable, presenting the outcome of the text-mining algorithm in a searchable form, in addition, ordering the documents by the relevancy to the keywords, and finally collecting the citations from all used documents in one text to copy it to the reference sections in the dissertation. Figure 14 shows the used technologies to develop the proposed solution.

Journal of Theoretical and Applied Information Technology
15th February 2022. Vol.100. No 3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

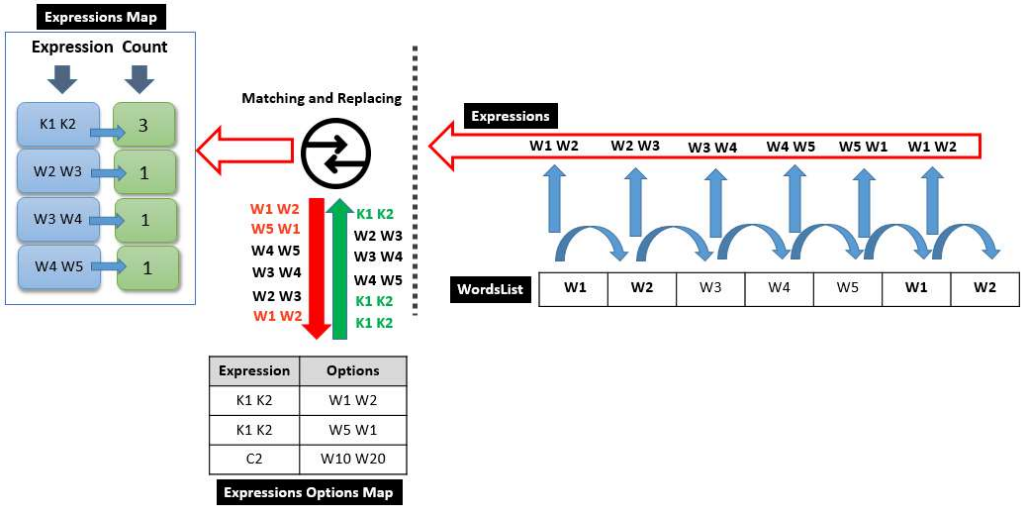


Figure 13. Matching and Replacing Expressions process

The system also produces some aggregations reports based on the metadata of the documents in a repository in visual forms as charts; the system manipulates and groups the metadata of the added documents to the system repository and presents them through charts, these charts can be used while conducting a systematic literature review.

Hence, this the researcher can focus on collecting articles with uncertain relevancy without spending time and effort on analysing and organizing the metadata and creating the aggregation reports manually, we will list the grouping reports provided by the system.

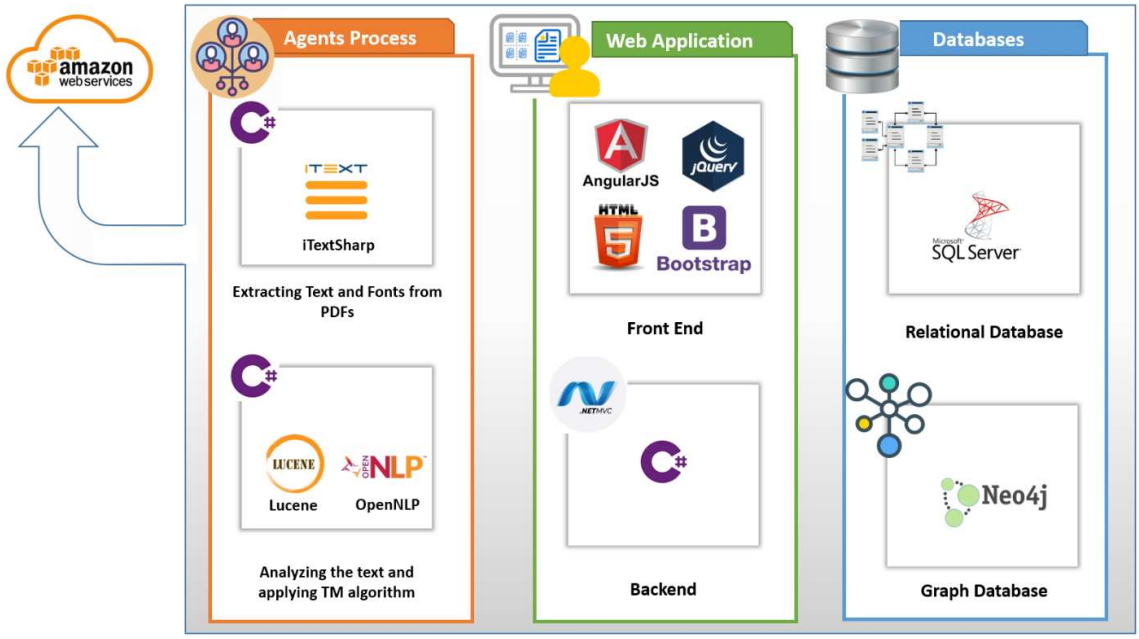


Figure 14 The Used Technologies To Develop the Proposed Solution

- Aggregation by issuing year: This chart groups the user provides the references of a particular repository by the year of issuing, against a list of ranges through the repository configuration, Figure 15 is an example of the grouping method against 5 ranges of years.

Journal of Theoretical and Applied Information Technology
15th February 2022. Vol.100. No 3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Figure 15 SLR Aggregation by issuing year

- Aggregation by Country: This chart groups the references of a particular repository by the Country, Figure 16 is an example of the grouping method.

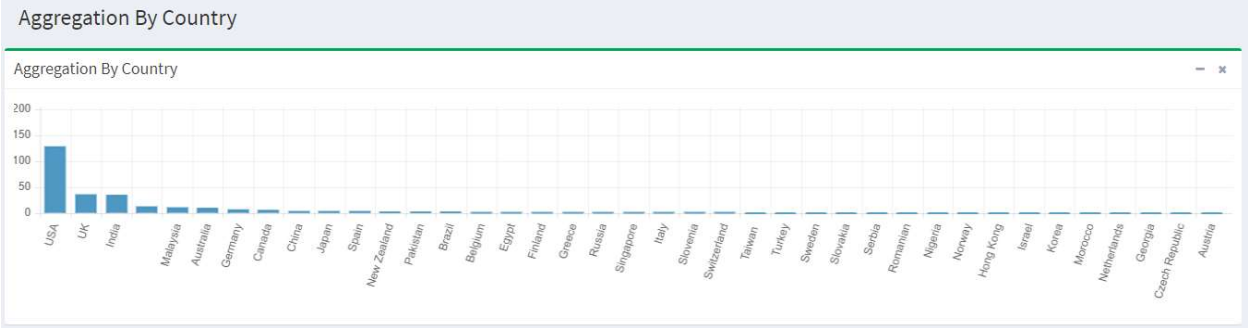


Figure 16 SLR Aggregation by Country

- Aggregation by Reference Type: This chart groups the references of a particular repository by the Reference Type, Figure 17 is an example of the grouping method.

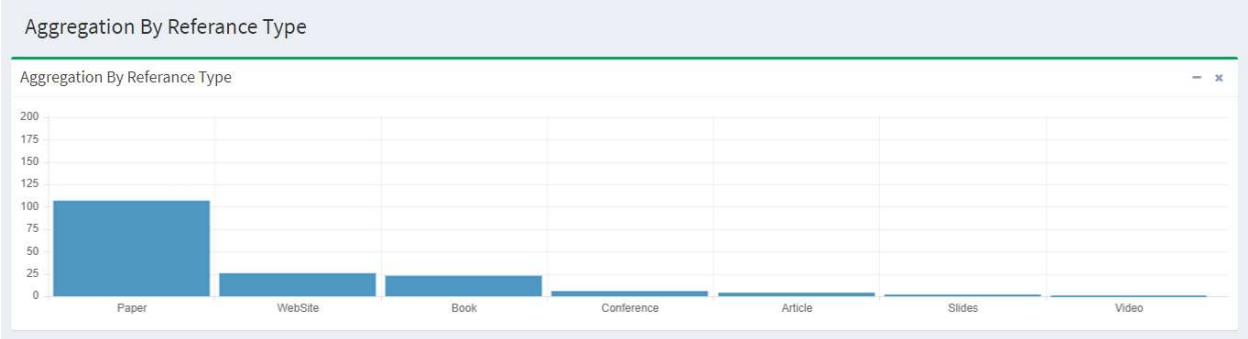


Figure 17 SLR Aggregation by Reference Type

- Aggregation by number of citations: This chart groups the user provides the references of a particular repository by the number of citation according to Google Scholar as per the date of inserting each document in the database, against a list of ranges through the repository configuration, Figure 18 is an example of the grouping method against 9 ranges of years.

5. TESTING AND VALIDATION

This section runs the document classification algorithm to parse testing sample studies to extract the keywords and the thresholds from each study and the expressions.

Figure 18 SLR Aggregation by number of citations

As mentioned earlier, the process of text mining, such as document classification, is considered a heavy process in terms of time and processing. However, we introduced the use of Multi-Agent as a proposal to improve the performance of the process and decrease the time. Hence, we run the algorithm through one and multiple agents in one machine and compare the time of the entire parsing process, starting from collecting the documents of studies from the user by the MAC (Multi-Agent Coach) until delivering the results of the parsing process to the DB agent. Then we run the algorithm in multiple machines (multi-nodes) simultaneously to see the positive impact of running the agents in multiple nodes on parallel in improving the parsing process's performance by decreasing the time of the operation.

We use SQS service (Queue Service in the cloud) from Amazon Web Service (AWS) on the cloud to push all 20 documents to the queue so the agents can pull the tasks (documents) from that queue to guarantee that one document will not be parsed more than one time from multiple agents.

Figure 19 shows the time consumed by each number of agents to parse the 20 documents in seconds, hence by comparing the times, we conclude that running the document classification algorithm through 4 agents in one machine decreases the total time of the process from 2283 seconds to 1523 seconds, means improving the performance approximately by 33.5%.

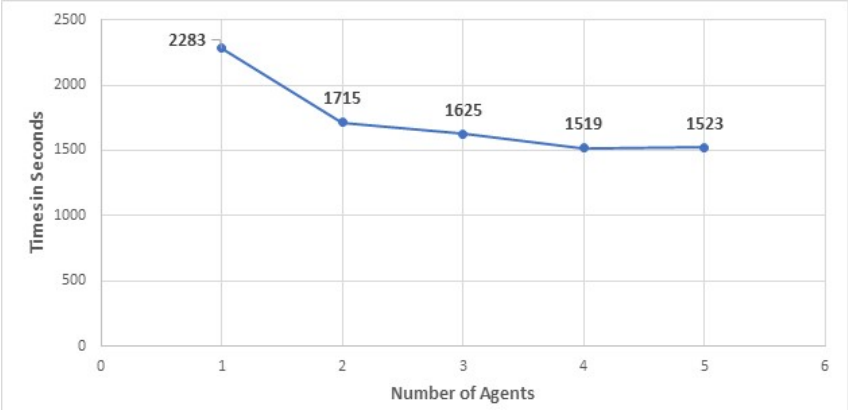


Figure 19 Performance Comparison - Running the algorithm by variant numbers of agents

However, as shown in Figure 20, running the algorithm through 5 agents and more did not improve the performance. The process took the almost same time, consequently, by keeping increasing the number of agents does not cause better performance because when the agents run simultaneously in one machine, they fight for the same OS resources, which means that increasing the number of agents dramatically and not thoughtful might grind the operating system.

We monitored the CPU of the machine during the parsing process and took screenshots from the CPU. As shown in Figure 21, we notice that increasing the number of agents working simultaneously causes increasing in consuming the machine's CPU accordingly, impacting other processes on the machine or the OS itself.

Journal of Theoretical and Applied Information Technology
15th February 2022. Vol.100. No 3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Figure 20 CPU monitoring during the Parsing Process

Many programming languages provide an encapsulated approach for managing the agents to facilitate creating the agents and avoid grinding the OS, so it creates the agents and assigns a task to them. Still, some jobs will execute simultaneously. The rest will queue up and wait their turn. We tested this approach, but that did not give any improvement compared with the 4 Agents since the agents were sharing the machine's resource. For example, if we have two documents (document 1, document 2) and run 1 agent to parse both of them, then run 2 agents to do the same, the single agent needed 23 sec to parse (document 1) and 33 sec to parse (document 2) but because the parsing both document using a single agent is done on the sequence, so the total time of the process is (23+33= 55 sec), now if we perform the same experiment using 2 agents to parse the same documents, document 1 needed 30 sec to get parsed, and document 2 needed 44 sec to get parsed, but because the agents parsed both documents on parallel, hence the entire process time was 44 sec as shown in Figure 16.

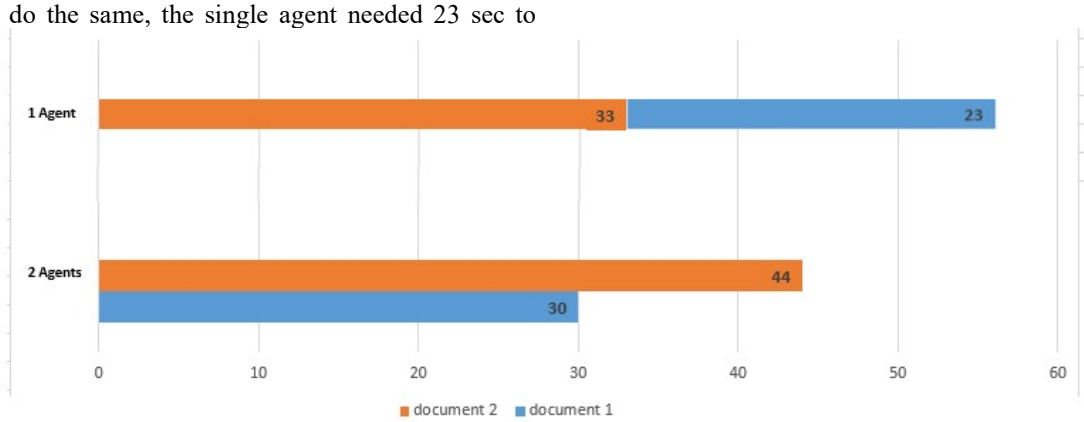


Figure 21 Parsing two documents using single-agent and two agents

We used the AWS EC2 service to issue three more instances of the same machine that we used in the previous section, run the four machines together to parse the testing sample (20 documents), and run four agents in each machine. Hence the number of agents is 12 agent as per this criteria. From Table 3, we notice that increasing the number of agents by using multi-agent within multiple machines on parallel can cause a massive improvement of the parsing process performance, where the time of the process decreased 1519 Sec to 350 Sec = 05:Min:50Sec.

While designing the document classification algorithm, we noticed that there the parsing process could not extract the text from some PDF files correctly, where three documents out of 200 (1.5%) of the documents gave invalid text since some of them might be built as images, not as a text. Hence, our parsing process can not extract any text from the images, where this process needs to use image processing algorithms and doing OCR, which is not available in the current parsing process. Hence, the parsing process gave valid results with 98.5%. Thus, we have 1.5% invalid results because of extracting the text from the PDF documents.

6. CONCLUSION AND FUTURE WORK

This study explored the use of AbDC in the systematic review process. The technology is implemented in screening the studies on behalf of the user to reduce the effort and time needed in the inclusion/exclusion stages in the systematic review process. Many document classification algorithms and factors affecting the text mining and document classification processes have been investigated. All the findings are combined. A new document classification algorithm is then concluded to help users exclude the irrelevant studies and order the relevant ones by the level of relevancy against desired topics. An agent-based model is proposed to accelerate the process and overwhelm the slowness of the document classification process.

Table 3 Parsing the testing sample using 4 Machines

Machine	The Parsed Documents	Time of the Process
Machine 1	Document_A.PDF Document_F.PDF Document_G.PDF Document_M.PDF	290 Sec
Machine 2	Document_B.PDF Document_L.PDF Document_O.PDF	284 Sec
Machine 3	Document_D.PDF Document_E.PDF Document_I.PDF Document_N.PDF Document_R.PDF Document_T.PDF	320 Sec
Machine 4	Document_C.PDF Document_H.PDF Document_J.PDF Document_K.PDF Document_P.PDF Document_Q.PDF Document_S.PDF	350 Sec

The use of software agents to measure the relevancy level and identify irrelevant studies from a set of studies is also demonstrated. The agent can calculate the weights of each study against desired topics and recommend the most relevant studies useful to the researcher. The algorithm is designed using a modern programming language as a backend engine to simulate the algorithm in actual studies. Moreover, a web-based systematic review tool is established to search amongst the extracted keywords to validate the designed document classification algorithm. Many features are provided to help the researchers in conducting the systematic review; these features include meta-data analysis, collection of meta-data and citation texts from a third party, creation of study repositories and study management and providing two types of searching within the repositories of studies (full-text and meta-data search). The results of this study in mitigating the systematic review process using AbDC have shown considerable success concerning the research objectives.

However, this study also shows some limitations and deficiencies. Nonetheless, these deficiencies do not compromise the significance of this research. The research scope is as follows: using document classification to reduce the required effort needed in the systematic review process overlays many discoveries that could be integrated into agents. This study also outlined some interesting areas that could be investigated in future work. Proposing a method to distinguish the Abstract and Keywords sections automatically from the studies can be useful to the parsing process, which can be run in two levels. The first round of screening involves only the Abstract and the Keywords sections, and the second round screens the entire study. The process can exclude the studies that do not show any kind of relevance in the first round of screening without going through the entire text for the first time because the abstract and keywords give an initial clue regarding the study. Thus, massive amounts of work and processing are saved, which, in turn, is reflected in the cost and the time. The proposed system serves many users working in various research and topics. Thus, the users upload the documents of the studies to the system along with the meta-data of the studies. The extracted keywords are saved in relational and graph databases. Therefore, a recommendation system can be built on top of the current system, where the researcher can flag a study as useful to his topic. Thus, the system can recommend some studies similar to the survey found useful based on the relations between the studies built in accordance with sharing some keywords in certain thresholds. This condition can be valuable to investigating documents based on the similarity to a submitted document because this issue is still under research. Efficient ways are still currently unavailable.

ACKNOWLEDGMENT

This work is sponsored by Universiti Tenaga Nasional (UNITEN) under the Bold Research Grant Scheme No. J510050002.

REFERENCES:

[1] Kontonatsios, G., Brockmeier, A.J., Przybyła, P., McNaught, J., Mu, T., Goulermas, J.Y. and Ananiadou, S., 2017. A semi-supervised approach using label propagation to support citation screening. Journal of biomedical informatics, 72, pp.67-76.

[2] Scells, H. and Zuccon, G., 2018, January. Generating Better Queries for Systematic Reviews. In SIGIR (pp. 475-484).

[3] Greenhalgh, T., 1997. How to read a paper: Papers that summarise other papers (systematic reviews and meta-analyses). Bmj, 315(7109), pp.672-675.

[4] Okoli, C. and Schabram, K., 2010. A guide to conducting a systematic literature review of information systems research.

[5] Fink, A., 2005. Conducting research literature reviews: From the internet to paper. Sage.

[6] Rychetnik, L. and Frommer, M., 2002. A schema for evaluating evidence on public health interventions: version 4. National Public Health Partnership (Australia).

[7] Bero, L.A., Grilli, R., Grimshaw, J.M., Harvey, E., Oxman, A.D. and Thomson, M.A., 1998. Closing the gap between research and practice: an overview of systematic reviews of interventions to promote the implementation of research findings. Bmj, 317(7156), pp.465-468.

[8] Cohen, A.M., 2008. Optimizing feature representation for automated systematic review work prioritization. In AMIA annual symposium proceedings (Vol. 2008, p. 121). American Medical Informatics Association.

[9] Cohen, A.M., Hersh, W.R., Peterson, K. and Yen, P.Y., 2006. Reducing workload in systematic review preparation using automated citation classification. Journal of the American Medical Informatics Association, 13(2), pp.206-219.

[10] Matwin, S., Kouznetsov, A., Inkpen, D., Frunza, O. and O'blenis, P., 2010. A new algorithm for reducing the workload of experts in performing systematic reviews. Journal of the American Medical Informatics Association, 17(4), pp.446-453.

[11] O'Mara-Eves, A., Thomas, J., McNaught, J., Miwa, M. and Ananiadou, S., 2015. Using text mining for study identification in systematic reviews: a systematic review of current approaches. Systematic reviews, 4(1), p.5.

[12] Han, J., Pei, J. and Kamber, M., 2011. Data mining: concepts and techniques. Elsevier.

[13] Feng, L., Chiam, Y.K. and Lo, S.K., 2017, December. Text-mining techniques and tools for systematic literature reviews: A systematic literature review. In 2017 24th Asia-Pacific Software Engineering Conference (APSEC) (pp. 41-50). IEEE.

[14] Cook, D.J., Mulrow, C.D. and Haynes, R.B., 1997. Systematic reviews: synthesis of best evidence for clinical decisions. Annals of internal medicine, 126(5), pp.376-380.

[15] Liberati, A., Altman, D.G., Tetzlaff, J., Mulrow, C., Gtzsche, P.C., Ioannidis, J.P., Clarke, M., Devereaux, P.J., Kleijnen, J. and Moher, D., 2009. The PRISMA statement for reporting systematic reviews and meta-analyses of studies that evaluate health care interventions: explanation and elaboration. PLoS medicine, 6(7), p.e1000100.

[16] Crawford, C. C., Boyd, C. C., & Jonas, W. B. (2015). Systematic reviews in practice. Alexandria, VA: Samueli Institute.

[17] Machi, L.A. and McEvoy, B.T., 2016. The literature review: Six steps to success. Corwin Press.

[18] Umnedu. 2019. CONDUCTING A SYSTEMATIC REVIEW. Health Sciences Libraries. [Online]. [30 July 2019]. Available from: https://hsl.lib.umn.edu/biomed/help/systematic-review

[19] Budgen, D. and Brereton, P., 2006, May. Performing systematic literature reviews in software engineering. In Proceedings of the 28th international conference on Software engineering (pp. 1051-1052). ACM.

[20] Khan, K., Kunz, R., Kleijnen, J. and Antes, G., 2011. Systematic reviews to support evidence-based medicine. Crc Press.

[21] macgill, M.A.R.K.U.S. (2019, 25 February 2019). What is a systematic review in research. [Weblog]. Retrieved 30 July 2019, from https://www.medicalnewstoday.com/articles/281283.php

[22] Nicholson, J., McCrillis, A. and Williams, J.D., 2017. Collaboration challenges in systematic reviews: a survey of health sciences librarians. Journal of the Medical Library Association: JMLA, 105(4), p.385.

[23] Mallett, R., Hagen-Zanker, J., Slater, R. and Duvendack, M., 2012. The benefits and challenges of using systematic reviews in international development research. Journal of development effectiveness, 4(3), pp.445-455.

[24] Subramainan, L., Mahmoud, M.A., Ahmad, M.S. and Yusoff, M.Z.M., 2016. An Emotion-based Model for Improving Students' Engagement using Agent-based Social Simulator. International Journal on Advanced Science, Engineering and Information Technology, 6(6), pp.952-958.

[25] Zhang, W.R. and Zhang, L., 2004. A multi-agent data warehousing (MADWH) and multi-agent data mining (MADM) approach to brain modeling and neurofuzzy control. Information Sciences, 167(1-4), pp.109-127.

[26] Wooldridge, M., 2009. An introduction to multi-agent systems. John Wiley & Sons.

[27] Vlassis, N., 2007. A concise introduction to multi-agent systems and distributed artificial intelligence. Synthesis Lectures on Artificial Intelligence and Machine Learning, 1(1), pp.1-71.

[28] Di Fatta, G. and Fortino, G., 2007, March. A customizable multi-agent system for distributed data mining. In Proceedings of the 2007 ACM symposium on Applied computing (pp. 42-47). ACM.

[29] Sycara, K.P., 1998. Multi-agent systems. AI magazine, 19(2), pp.79-79.

[30] Balaji, P.G. and Srinivasan, D., 2010. An introduction to multi-agent systems. In Innovations in multi-agent systems and applications-1 (pp. 1-27). Springer, Berlin, Heidelberg.

[31] Maseleno, A., Tang, A. Y. C., Mahmoud, M. A., Othman, M., Saputra, S., & Muslihudin, M. (2017). Fuzzy AHP method to determine headache types based on symptoms.

[32] Mahmoud, M. A., & Ahmad, M. S. (2016, August). A prototype for context identification of scientific papers via agent-based text mining. In 2016 2nd International Symposium on Agent, Multi-Agent Systems and Robotics (ISAMSR) (pp. 40-44). IEEE.

[33] Mahmoud, M. A., Ahmad, M. S., Ahmad, A., Yusoff, M. Z. M., & Mustapha, A. (2011, July). Norms detection and assimilation in multi-agent systems: a conceptual approach. In Knowledge Technology Week (pp. 226-233). Springer, Berlin, Heidelberg.

[34] Subramainan, L., Yusoff, M. Z. M., & Mahmoud, M. A. (2015, August). A classification of emotions study in software agent and robotics applications research. In 2015 International Symposium on Agents, Multi-Agent Systems and Robotics (ISAMSR) (pp. 41-46). IEEE.

[35] Subramainan, L., Mahmoud, M. A., Ahmad, M. S., & Yusoff, M. Z. M. (2016). An Emotion-based Model for Improving Students' Engagement using Agent-based Social Simulator. International Journal on Advanced Science, Engineering and Information Technology, 6(6), 952-958.

[36] Mahmoud, M., Ahmad, M. S., Mostafa, S., & Subramainan, L. (2020). How new individuals behave in a heterogeneous community: a computational approach to norm assimilation using agent-based systems. Journal of Systems Science and Complexity, 33(4), 849-881.

[37] Gurunathan, M., & Mahmoud, M. A. (2020). A review and development methodology of a lightweight security model for IoT-based smart devices. International Journal of Advanced Computer Science and Applications, 2, 125-134.

[38] Mahmoud, M. A., & Grace, J. (2019). A generic evaluation framework of smart manufacturing systems. Procedia Computer Science, 161, 1292-1299.

[39] Mahmoud, M. A., Ahmad, M. S., & Mostafa, S. A. (2019). Norm-based behavior regulating technique for multi-agent in complex adaptive systems. IEEE Access, 7, 126662-126678.

[40] Mahmoud, M. A., Ahmad, M. S., Yusoff, M. Z. M., & Mostafa, S. A. (2018, February). A regulative norms mining algorithm for complex adaptive system. In International Conference on Soft Computing and Data Mining (pp. 213-224). Springer, Cham.

[41] Subramainan, L., Mahmoud, M. A., Ahmad, M. S., & Yusoff, M. Z. M. (2017, June). A simulator's specifications for studying students' engagement in a classroom. In International Symposium on Distributed Computing and Artificial Intelligence (pp. 206-214). Springer, Cham.

[42] Mahmoud, M. A., Ahmad, M. S., Yusoff, M. Z. M., & Mustapha, A. (2014, December). Norms assimilation in heterogeneous agent community. In International Conference on Principles and Practice of Multi-Agent Systems (pp. 311-318). Springer, Cham.

[43] Mostafa, S. A., Ahmad, M. S., Ahmad, A., Annamalai, M., & Mustapha, A. (2014). A dynamic measurement of agent autonomy in the layered adjustable autonomy model. In Recent developments in computational collective intelligence (pp. 25-35). Springer, Cham.

[44] Mostafa, S. A., Darman, R., Khaleefah, S. H., Mustapha, A., Abdullah, N., & Hafit, H. (2018, June). A general framework for formulating

adjustable autonomy of multi-agent systems by fuzzy logic. In KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications (pp. 23-33). Springer, Cham.

[45] Mostafa, S. A., Ahmad, M. S., Ahmad, A., Annamalai, M., & Gunasekaran, S. S. (2016, August). A Flexible Human-Agent Interaction model for supervised autonomous systems. In 2016 2nd International Symposium on Agent, Multi-Agent Systems and Robotics (ISAMSR) (pp. 106-111). IEEE.

[46] Mostafa, S. A., Ahmad, M. S., Annamalai, M., Ahmad, A., & Gunasekaran, S. S. (2015). Formulating dynamic agents' operational state via situation awareness assessment. In Advances in Intelligent Informatics (pp. 545-556). Springer, Cham.

[47] Mostafa, S. A., Ahmad, M. S., Ahmad, A., & Annamalai, M. (2013, December). Formulating situation awareness for multi-agent systems. In 2013 International Conference on Advanced Computer Science Applications and Technologies (pp. 48-53). IEEE.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195