Abstract

Big data analytics is gaining popularity for enterprises in optimizing their business processes ranging from retailers, supply chains, to online shopping stores. Existing practical raw data are far from usable to achieve the goal. Therefore, a good data pre-processing approach is required and that is a key step to success. We study on the effectiveness of data pre-processing with respect to the optimal business process based on a real world database. Our methodology involves natural language processing and semantic analysis. Our key goal is to study appropriate methods with big data analysis techniques that can handle errors, ambiguity, and repeated descriptions caused by human languages. In this thesis, we study a simple language similarity checking in order to optimize the databases used in daily business process. A logical representation system is built from our database as a proof of concepts.

Keywords：Big Data Analytics, Databases, Logical representation

ACKNOWLEDGEMENTS

I would like to thank my Adviser, Dr. Dan Lo and Dr. Po Yueh Chen, for Doctors' guidance and encouragement over the course of my study at Kennesaw State University, Marietta, Georgia and National Changhua University of Education, Taiwan. I am very appreciated for Doctors' experience and assistance in completing this thesis.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

Introduction

Recent decades, big data analytics has become one of the hot topics around the world [11]. Due to the growing capacity of computer system's storage and the support of cloud storage [12], even the personal computing systems could have storages of at least terabytes. It reduces the limitation and derives several related field popular fields [13], such as machine leaning and Artificial intelligence. The larger storage means that we can have more data samples and it might let the training model perform better without bias, but it also might raise the difficulty to handle the big database [14] [22]. Maintaining the database also becomes one of the important tasks in an analysis process.

For normal statistical data, a lot of methods for the specific problem have been proposed, such as, Recurrent neural network (RNN) [1] and Long Short-Term Memory (LSTM) [1]. In computer vision or image processing related problems, convolutional neural network (CNN) [15] has been successfully employed to detect the important features in images. When it comes to natural language problems [16], they might be harder than normal data. RNN is suitable for streaming data, may be used in the field like handwriting or speech recognition. LSTM method can be used to predict the next characters, according to the historical information in the training data. Those methods are training with feedback, but seems not relating to the meaning of the content.

Although the machine learning technique with hardware support is very powerful, natural language processing with semantic analysis is still a big problem [17]. The method mentioned above, is first to tokenize the sentence into words [28], and then to find the relationship among each elements to produce the final result. The fundamental reason for the difficulty is “human languages are ambiguous [18].” Human languages are affected by various factors from people to people [25]. As a result, having a rule in datasets might be useful to keep datasets tidy.

For new created databases, we may set a rule or standard in order to keep the dataset being consistent and easy to handle. If the natural language database had already been built in a large size, it would have been hard to make some changes afterwards. Our goal in this research is to examine the feasibility of existed methods, one of which was proposed by Martínez-Gómez, *et al.* [2], on the real tech company’s database, which processed English articles. Their system can let user transfer documents in natural language into high order logic representations. The authors in [3] shows that after transferring English to logic representations, we can utilize logic operations to reduce the data size, which will help us handle the datasets better.

Unlike using normal, non-logic expressions, using formal logical expressions could have multiple benefits for users to handle the database. In [3], the authors pointed out that, the logic expressions with standards allow users to implement many logical reductions, and may allow users to have more flexibility to do much more operations than those in their original format [20]. First order logic(FOL) [19] has two important components: Terms, and Formula. Terms can be variables or contents and Formula shows the relation among them. FOL allows us to represent a complex, large content in a simple formula. Some rule like “Unification” can be used to reduce the database

size by finding some similarity in different representations. “Resolution Proofs” can be used to do simple inference in First order logic [24].

In the next chapter, we will introduce some related works. In Chapter III, we will introduce our experimental settings. In Chapter IV, we will report the experimental results and analyses followed by a discussion. Lastly, we will give the conclusion of our study and future work in the last Chapter.

CHAPTER II

Related Work

In the paper [4], the authors brought the big data analytic technique into the real world applications. They tried to help a tech company, Intelliteach, to make their business process automatically by applying the Big Data analytics [21]. In their experiments, they used four decisive attributes, which can correspond to the problem category of a ticket, to training the machine learning model for testing the possibility of automation. As a result, they found that not only different machine learning algorithms affects the hit rate, which is a measure of the automatic process, but also the number of samples also affects the result. In their dataset, due to the variety of the issue's categories, the hit rate will go down if they fed in more data into training. Therefore, we have to pay more attention on the trade-off between data completeness and complexity.

Intelliteach is a typical IT company in the U.S., and the data base is from its helping desk services systems, which is a collection of technical problems from its customers, typically law firms, and the solution for that issue will be responded, according to their database. In [4] Lo. *et al.* used four of the attributes in the dataset for the machine learning model. In our study, we also used this dataset, but different table. Those are "KbArticles", which is a solution knowledge database, and "Categories", which contains the entire technical problem category, and the "KbArticleCategories", which indicates the relationship between "KbArticles" and "KbArticleCategories". We will discuss

about the database detail in the next section.

In paper [2], the authors proposed an easy-to-use system that transforms human natural language into a logical representation, and the output is telling that if the last sentence can be an inference with the previous sentence. Input is text file with one sentence per line, and then transformation output is a CCG parsing tree. Next step is using CCG tree and pre-created semantic templates to get the semantic composition. Lastly, using the prover can give the final answer, whether the last sentence can be the inference or not.

According to [5] and [6], we can treat Combinatory Categorial Grammar(CCG) as a different method to present a human language. CCG will classify every word tokens separately, language token into different categories, and it could be normal element or the function, which can be seen as a relationship between different element. After setting all the corresponding representation, CCG uses leftmost or rightmost notation to convert the symbolic element into a meaningful and complete representation.

CHAPTER III

Experimental Setup

3.1) DATABASE DETAIL

Microsoft SQL server(MSSQL) [7], was developed by Microsoft, and it's first version was launched in 1989. MSSQL is a relational database, which stores data in a predetermined relationship and organizes the data into tables with multiple columns and multiple rows. Each column represents a unique attribute and datatype, and the rows represent a set of object, with their own unique keys. Since MSSQL is a relational database, users can use the SQL language as a query to request data or perform some operations in the database. With that operations, users can access the specific data without rearranging the database.

In our experiment, we used MSSQL local sever to handle the database. The original database from the company is in ".bak" file. We first restored the file into the MSSQL server, and then request the data using SQL language. When we find the exact data we want, we use the SQL server to extract the data and storage then into ".csv" files, which can be used in python, and read the data outside the database.

The database we used in this paper is same as in [4] by Lo. *et al.*, but we are using different tables. As we mentioned above, they used the attributes in "Ticket" table. In this work, we are using the "ArticleHTML" column and "ArticleTitle" column in the

knowledgebase table. There is another table called “KbIdtoCateId“, which can indicate the category of the data in knowledgebase. We used Natural Language Toolkit (NLTK) [8] with the data in “AticleTile” column, which are in the same category, to check the similarity of the data. We used the data in “ArticleHTML” table, which contain the detail of the issue situation and possible solutions, to implement the ccg2lambda system [2]. The relationship of the database is showing in Figure 1.

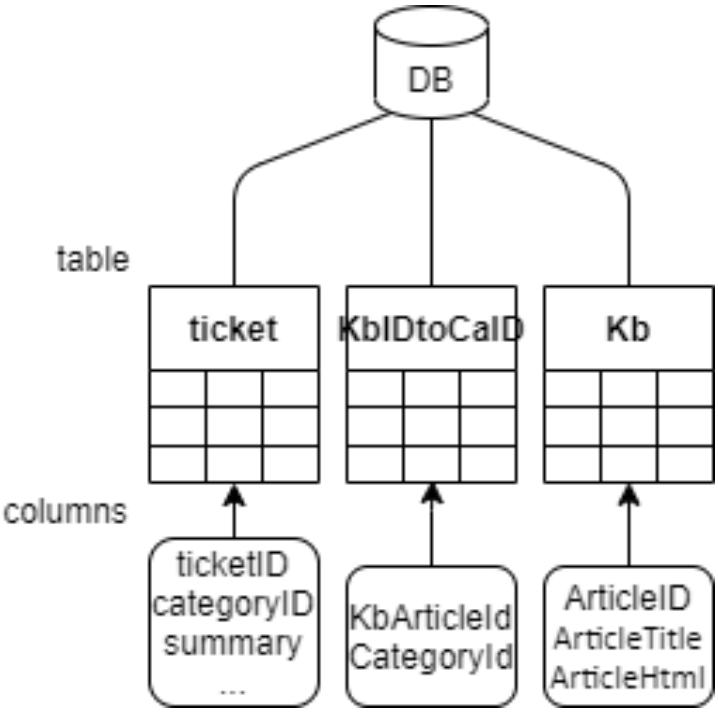

Figure 1. In Intelliteach IT company database, we used the data in KB table: knowledgebase, which contains the issues and the possible solutions. KbidtoCaid table indicates which KB is in the same category.

3.2) EXPERIMENT DETAIL

3.2.1) Similarity Checking

Before transforming data, we want to understand more about the database. So we did the similarity checking between the “ArticleTitle” and “CategoryName” column. We will introduce the data statue information and the environment used in this experiment.

3.2.1.1) Data Status

We found three relationships between Knowledge Base and Category with SQL server:

Case1: One category matches more than one KB

In this case, data in one category of problems match multiple solution knowledge bases. We search matched data and export the data into Excel via SQL server manager studio. (Table 1)

There are more than one category matches with many KB's. So we select the Kb data belonging to category(ID 17756) to do a similarity comparison, which has the most number of KB data (997).(Figure 2)

KB id	CategoryId	Number
91091	17756	997
91564	127	310
47341	76204	292
84326	85425	271

Table 1. None-one on one matched data example (top Four Data) KBid represents the first Knowledge id in this category.

	CategoryId	KbArticleId	KBinkbAform	ViewCount	ArticleTitle
1	17756	41301	41301	38	PF - Equitrac/Billback frozen or no com
2	17756	41635	41635	35	PF- iPay - Cannot Print Pay Statement
3	17756	44343	44343	47	INF- Equitrac Home Page
4	17756	44997	44997	52	PF - LookUp Precision will not login to
5	17756	45244	45244	39	Internetiff 7.1 Pro Save to PDF
6	17756	46491	46491	59	INF - Non-Core Apps: Axcelerate Appli
7	17756	40527	40527	54	Ensuring that Endpoint Protection is act

Figure 2. Part of the data with Category ID 17756.

Case2: 1 Category matches 1 KB

The second case is the ideal data type, KB and category are matched one-to-one. We export these data for a control group, make sure if this case has the last similarity. (Table 2)

KB id	CategoryId	Number
39106	10	1
13192	25	1
26712	49	1
26127	50	1

Table 2. One on one matched data example (top four data).

Case3: One KB matches more than one category

We also found that there are plenty of different problem Categories linked to the same KB. In this Case, we export KB (ID63539) which has the most number of Category data, and calculate the similarity between the “CategoryName” column. (Figure 3)

	KbArticleId	CategoryId	CateidInCateform	CategoryName
1	63539	66665	66665	Software Services
2	63539	66666	66666	Hardware
3	63539	66667	66667	Admininstrative Services
4	63539	66668	66668	CD/DVD
5	63539	66669	66669	Conference Services
6	63539	66676	66676	Desktop
7	63539	66677	66677	CD/DVD Drive

Figure 3. Part of the data with knowledge ID 63539

3.2.1.2) Natural Language Toolkit (NLTK)

NLTK [8] is a natural language processing library written in Python. It is a well-built library, and we can use the library functions by just downloading the package and importing the functions we need. We use this toolkit to clean some extra notation,

noisy characters, and the stop words in the data, including using tokenizer to tokenize the clean textures, and using lemmatization to transform inflection form back into the basic form.

3.2.1.3) Cosine Similarity in Google Colab

According to [10], Vector Cosine Similarity (1) is a common and basic technique in text mining problem to check the similarity between two segments. Because it is easy to use and easy to understand, we decided to use cosine similarity to compare data. The method is simple: we pick the data in “Articletitles” and “CategoryName” and compare them. The time complexity of the comparison is O(n^2). We recorded highest, average, and smallest similarity, total comparison time, and during time to indicate the status of the data. We used the “Articletitles” column instead of the “ArticalHTML” column, which contains more information, because the comparison process is time-consuming and not suitable for long contexts.

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{||A|| \, ||B||} \qquad (1)$$

3.2.2) Ccg2lambda System Applying

3.2.2.1) Data

We used the first three “ArticleHTML” columns in category ID no.1. The row data in the column contained the information and the HTML notations, therefore, we did some data pre-processing. First, we removed the HTML tags, such as “<div/” etc. Second, because the ccg2lambda system had input format, we changed the input file with one sentence per line. Third, input the data and check the result.

3.2.2.2) Environment

For this implement, we have to change the working environment, because the original system has some Linux operations. We set up an Ubuntu desktop version 1.8 [23] with 2 GB memory and 20 GB storage. We set up the system with a virtualbox 6.1.18 in Windows 10.

3.2.2.3) System

As the mention above, ccg2lamda system [2] was used to transfer the text and check whether the inference is correct or not. In our implement, we did not do that yet, because our final goal is transform the non-logic language into logical expressions. In this study, we stop execution after the logical tree and logical expression are constructed, and we get the logical expression from the result.

CHAPTER IV

Results and Discussion

4.1) Similarity checking Result

In the result, we use Average_similarity, Largest_similarity, Smallest_similarity, and total comparison number, four parameters to determine the data's status.

Average similarity is the average sentence similarity in that specific case, and this can be used to compare between two data sets. Largest and smallest similarity indicates the status between just two data entries in the data set.

The total number shows that if all the data are useable, the total number of compare should be $(\text{data number} - 1) \times (\text{data number})/2$. If this number does not match we know that some data are empty.

Case 1: One category matches more than one KB

We analysis KB title, which has same Category ID 17756. Total number of the data is 997. The experiment result: total compare number of data is 378885.0, average_similarity is 0.03341, largest_similarity is 1.0, and smallest_similarity is 0.0.

Total comparison number should be 496,506, but we get 378,885. It indicates that that some of KB belonging to this category ID are empty.

The largest similarity was 1.0. This indicate that at least two KB's were exactly the same in this data form. We definitely can reduce the size of this data set by remove the same one.

Case2: One category matches one KB

In this case, the total number of the data is 10819. The result we got: the total compare number of data was 58519971.0, the Average_similarity was 0.00896, the largest_similarity was 1.0, and the Smallest_similarity was 0.0.

In this one on one match case, it has the lowest average similarity than the others. The total comparison number should be 58,519,971, and we get 58,519,971, So there is no element that is empty.

However, we still got 1.0 largest_similarity. It shows that there are some with the same KB.

Case3: One KB matches more than one Category

In the third case, there are 456 numbers of data. The results are: The total compare number of data was 103740.0, the Average_similarity was 0.01762, the largest_similarity was 1.0, and the Smallest_similarity was 0.0.

The comparison number we get is 103,740, which is exactly what it should be. So there is no empty slot. The 1.0 largest similarity shows the duplicate existed.

The overall data are illustrated in Table 3. In theory, under normal circumstances, an issue category should exactly correspond to one database. So it will be the most efficient. However, in real cases, an IT tech issue could have multiple situations. According to the result, these datasets all have at least two article titles that are the same as the other. Duplicate data will make the machine confuse and slow down the execution speed. By reducing the duplicated data, we may lower the similarity and the total data size. After optimization, it might increase the classification hit rate, and we might have the chance to improve the business process.

	Case 1	Case 2	Case 3
Largest Similarity	100%	100%	100%
Average Similarity	3.341%	0.896%	1.763%
Smallest Similarity	0%	0%	0%
Total data number	997	10,819	456
Total comparison time	378,885	58,519,971	103,740
Total comparison should be $(\text{data number} - 1) * (\text{data number})/2$	496,506	58,519,971	103,740
Processing time	99 sec	4.32 hours	22 sec

Table 3. Overall similarity in three case.

4.2) Ccg2lambda System Applying result

4.2.1) Data pre-processing

4.2.1.1) Remove HTML tags and some irrelevant notations in the text file

The row data in the database contains some unused notations, which have to be removed. Using the regular expression library “re” [26] built in the python, we can import that and re-compile the program to get clean data as reported in (Figure 4).

	ArticleHtml	new
0	Problem: In Office XP and 2003, Wo...	Problem: In Office XP and 2003, Word, Excel an...
1	Problem: When opening a deltaview ...	Problem: When opening a deltaview document if ...
2	Summary: If a user ever wants to c...	Summary: If a user ever wants to create an aut...
3	Summary: We all know how to right ...	Summary: We all know how to right click a file...
4	How to print a file list. The files...	How to print a file list. The files must be in...
..	...	...
93	<p> Issue: You have enabled an add ...	Issue: You have enabled an add in in an MS Off...
94	Issue: Change matter name spelling...	Issue: Change matter name spelling in Sos Conn...
95	How to apply ﷺFull Rightsﷺ to Export Email/Exc...	How to apply Full Rights to Export Email/Excel...
96	ISSUE: User needs to find hidden dial...	ISSUE: User needs to find hidden dialog box. S...
97	Issue:<br...	Issue: When someone needs their Transcription ...

Figure 4. Left-hand side shows the original text data; Right-hand side is the data after removing the HTML tags

4.2.1.2) Making the data one sentence per line

We did this part manually, that is due to some sentences without correct punctuations, It will make the sentence in a strange format. In Figure 5, there is no punctuation between last two sentences "...Office applications To have...", and that will make machine not easy to make into different line.

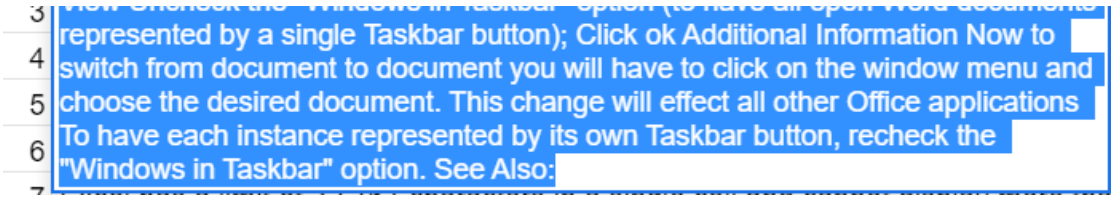

Figure 5. First Article HTML cleaned data in Category No.1

4.2.2) Error Message

When we run the system with our data, we receive some error messages.

1) input file does not match the system

In our data, there are some extra words in sentences (refer to Figure 4 left hand side). For instance, “Problem:, Issue..” etc. Those headers will make this system pop up error messages. To solve this, we remove the extra header in those file, and then the system run correctly.

However, this situation pointed out that the input of this system might have a hard requirement. If we want to automate the process, further data pre-processing is required, or system might be modified a little bit.

2) lambda function not existed

Beside the input format error, we also met another type of errors, which is from the template error. In this system [2], the author simplifies the original semantic template [27], which is used to construct a semantic combination for semantic understanding by Bos, J., *et al*. The author [2] mentioned that this reduction was for

generate more precise result.

In our experiment, however, our context seemed to be too complex, we received the error message in (Figure 6). In short, this message indicated that there was no semantic template match the output of CCG derivations. As a result, the system cannot generate the semantic representation.

In order to fix that, fortunately, the author provided the original template, which contain more complete semantic templates without reductions. After we change the template file, the whole process worked successfully.

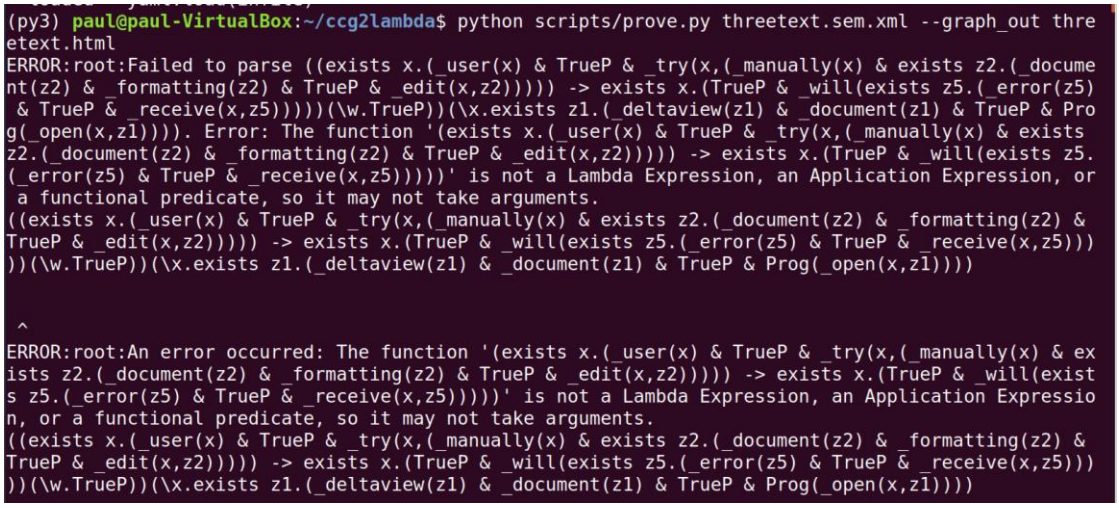

Figure 6. lambda function error message.

4.2.3) Result

Since the result was generated with CCG parsing tree, and the graphics were too large, we only picked a small part of the text.

The result of the input “The Number has to be from 1 to 600 error.” is “exists x.(_number(x) & TrueP & _have(x,exists z2.(_600(z2) & _error(z2) & TrueP & _to(x,z2) & _be(x) & exists z1.(_1(z1) & TrueP & _from(x,z1))))”. (Figure 7)

exists x.(_number(x) & TrueP & _have(x,exists z2.(_600(z2) & _error(z2) & TrueP & _to(x,z2) & _be(x) & exists z1.(_1(z1) & TrueP & _from(x,z1)))))

Figure 7. one of the logical representation result.

The result of the input text “Reason : The reason is that the document is still in Deltaview document format until the formatting is cleared .” we got “\ F1 F2.(exists x.(_reason(x) & F1(x) & F2(x)) & exists x.(_reason(x) & TrueP & _be(x,_until(exists x.(_formatting(x) & TrueP & _clear(x)),exists x.(_document(x) & TrueP & _still(x) & exists z1.(_deltaview(z1) & _document(z1) & _format(z1) & TrueP & _in(x,z1)))))))”.

The result of the input “Resolution : Clear the formatting of the specified text first by pressing the shortcut CTRL + Spacebar .” we got “\X0 Q.Q(\w.TrueP,\x.(_first(x) & _formatting(\F1 F2.exists z5.((z5 = _resolution) & exists z1.((z1 = _clear) & TrueP & _:(_resolution,_clear)) & F1(_resolution) & F2(_resolution)),\z5.(exists z2.(_specify(z2) & _text(z2) & TrueP & _of(z5,z2)) & X0(z5)),\F1 F2.F2(x)) & exists z3.(_shortcut(z3) & _ctrl(z3) & _+(z3) & _spacebar(z3) & TrueP & Prog(_press(x,z3)))))”.

We successfully convert a natural language sentence to a logical representation, but for large databases, there are some points that must be paid more attention to. First, the transferring processing should not involve too many human operations. In our data pre-processing stage, in order to match the input format, we did a lot of manual operations, which is not efficient for maintaining big data. Second, in our experiment, we got a lot of errors due to the lack of semantic templates. To create a complete semantic template file, which is known as the representation rule, for different databases that are efficient and necessary. Third, according to our result, the outputs are larger than the original inputs. To store these outputs and the semantic template, the storage utilization should be carefully considered in the system [22].

CHAPTER V

Conclusion and Future Work

In this thesis, we demonstrated the feasibility of using the ccg2lambda system to convert real-world text data in English into logical representations. Although too many manual operations are involved, the efficiency of the whole process is not high enough, but the result is marginal. We believe that the logical representation has a wide possibility than then non-logical representation.

For future work, two directions should go further. First, more efficient process from data pre-processing to the whole transferring process is necessary. There are many related and established systems, which also can transfer the non-logical data. Second, it is a must to study logical data reduction and rule building for the future data. In order to reduce the burden of the database, how to increase efficiency and reduce the maintenance cost of the database remains to be studied.

References

[1]. Sherstinsky, A. (2020). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena*, (vol.*404*, pp. 132306).

[2]. Martínez-Gómez, P., Mineshima, K., Miyao, Y., & Bekki, D. (2016, August). ccg2lambda: A compositional semantics system. In *Proceedings of ACL-2016 System Demonstrations* (pp. 85-90).

[3]. Ayorinde, I., & Akinkunmi, B. (2013). Application of first-order logic in knowledge based systems. *Afr J. of Comp & ICTs*, *6*, 45-52.

[4]. Lo, D., Tiba, K. K., Buciumas, S., & Ziller, F. (2019, July). An Emperical Study on Application of Big Data Analytics to Automate Service Desk Business Process. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)* (Vol. 2, pp. 670-675). IEEE.

[5]. Steedman, M., & Baldridge, J. (2011). Combinatory categorial grammar. *Non-Transformational Syntax: Formal and explicit models of grammar*, 181-224.

[6]. Baldridge, J., & Kruijff, G. J. M. (2003, April). Multi-modal combinatory categorial grammar. In *10th Conference of the European Chapter of the Association for Computational Linguistics*.

[7]. Nokrach, A. (2018). Comparing the databases MSSQL and MongoDB for the web-based environment Ozlab.

[8]. Loper, E., & Bird, S. (2002). Nltk: The natural language toolkit. *arXiv preprint cs/0205028*.

[9]. Bisong, E. (2019). *Building machine learning and deep learning models on Google cloud platform* (pp. 7-10). Berkeley: Apress.

[10]. Li, B., & Han, L. (2013, October). Distance weighted cosine similarity measure for text

classification. In *International conference on intelligent data engineering and automated learning* (pp. 611-618). Springer, Berlin, Heidelberg.

[11]. Kambatla, K., Kollias, G., Kumar, V., & Grama, A. (2014). Trends in big data analytics. *Journal of parallel and distributed computing*, *74*(7), 2561-2573.

[12]. Wu, Y., Lyu, Y., & Shi, Y. (2019). Cloud storage security assessment through equilibrium analysis. *Tsinghua Science and Technology*, *24*(6), 738-749.

[13]. Siddiqa, A., Karim, A., & Gani, A. (2017). Big data storage technologies: a survey. *Frontiers of Information Technology & Electronic Engineering*, *18*(8), 1040-1070.

[14]. Manning, A. (2015). Maintaining Your Database. In *Databases for Small Business* (pp. 167-172). Apress, Berkeley, CA.

[15]. Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, *61*, 85-117.

[16]. Gharat, A., Tandel, H., & Bagade, K. (2019). Natural language processing theory applications and difficulties. *IJTSRD*, *3*(6), 501-503.

[17]. Maulud, D. H., Zeebaree, S. R., Jacksi, K., Sadeeq, M. A. M., & Sharif, K. H. (2021). State of art for semantic analysis of natural language processing. *Qubahan Academic Journal*, *1*(2), 21-28.

[18]. Maurya, H. C., Gupta, P., & Choudhary, N. (2015). Natural language ambiguity and its effect on machine learning. *International Journal Of Modern Engineering Research*, *5*, 25-30.

[19]. Gergely, T., & Úry, L. (2012). *First-order programming theories* (Vol. 24). Springer Science & Business Media.

[20]. Walter, B., Hammes, J., Piechotta, M., & Rudolph, S. (2017, September). A formalization method to process structured natural language to logic expressions to detect redundant specification and test statements. In *2017 IEEE 25th International Requirements Engineering Conference (RE)* (pp. 263-272). IEEE.

[21]. Jha, M., Jha, S., & O'Brien, L. (2016, June). Combining big data analytics with business process using reengineering. In *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)* (pp. 1-6). IEEE.

[22]. Iqbal, M., Kazmi, S. H. A., Manzoor, A., Soomrani, A. R., Butt, S. H., & Shaikh, K. A. (2018, March). A study of big data for business growth in SMEs: Opportunities & challenges. In *2018 International conference on computing, mathematics and engineering technologies (iCoMET)* (pp. 1-7). IEEE.

[23]. Helmke, M. (2018). *Ubuntu Unleashed 2019 Edition: Covering 18.04, 18.10, 19.04*. Addison-Wesley Professional.

[24]. Ao, Z., & Lambert, D. (2012, July). Semantic inference by first order logic. In *2012 15th International Conference on Information Fusion* (pp. 204-210). IEEE.

[25]. Tamilselvi, J. J., & Gifta, C. B. (2011). Handling duplicate data in data warehouse for data mining. *International Journal of Computer Applications*, *15*(4), 7-15.

[26]. Chapman, C., & Stolee, K. T. (2016, July). Exploring regular expression usage and context in Python. In *Proceedings of the 25th International Symposium on Software Testing and Analysis* (pp. 282-293).

[27]. Bos, J., Clark, S., Steedman, M., Curran, J. R., & Hockenmaier, J. (2004). Wide-coverage semantic representations from a CCG parser. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics* (pp. 1240-1246).

[28]. Le, Q., & Mikolov, T. (2014, June). Distributed representations of sentences and documents. In *International conference on machine learning* (pp. 1188-1196). PMLR.