Volume: 8 Issue: VI Month of publication: June 2020
DOI: http://doi.org/10.22214/ijraset.2020.6391

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue VI June 2020- Available at www.ijraset.com

SOC Estimation and Automation of Battery Test Specification Sheet using NLP

Manish Kumar Mawatwal[1], Aayush Mohanty[2], Dr Anitha G.S.[3]
[1, 2, 3]RV College of Engineering, Electrical and Electronics Engineering, Bengaluru, India

***Abstract:** The use of automobiles is increasing at a rapid pace. The number of electric vehicles is a major chunk of this. People's safety is a major concern and there is a need to safely test the batteries of automobiles both at the physical level and software level. Software level testing requires manual work on different testing platforms, hence this research paper aimed to automate the process of creating corpus from the test specification sheet, to improve the overall safety and reliability of the behavior of battery and to bring efficiency in time.*

Estimating the State of Charge for each cell is very important for measuring the available energy and vehicle's range. In every system which includes a battery for improved battery control, SOC estimation is very essential. The coulomb counting method for the SOC estimation is achieved by developing electrochemical model-based estimation in MATLAB software for 3.6V rechargeable lithium nickel manganese cobalt (LiNMC) cell.

The methodology requires an in-depth understanding of Natural Language Processing including its features like Stemming, Vectorization, etc., and some basic idea of Artificial Intelligence (AI).

***Keywords:** Electric Vehicle, Battery Management System, Original Equipment Manufacturer, State of Charge, Machine Learning, Natural Language Processing, Coulomb counting*

I. INTRODUCTION

A. Battery Management System

Battery management system is an electronic controller that tracks and controls rechargeable batteries charging and discharging, regulates load / discharge levels based on load requirements, cell voltage of individual cells, current and temperature measurements (average temperature), and estimates battery State of Charge (SOC), power, impedance, etc. Figure 1 shows the BMS block diagram. Apart from this a BMS monitors various items, such as:

1) Depth of discharge (DOD)
2) State of Health (SOH)
3) State of Power (SOP)
4) State of Safety (SOS)

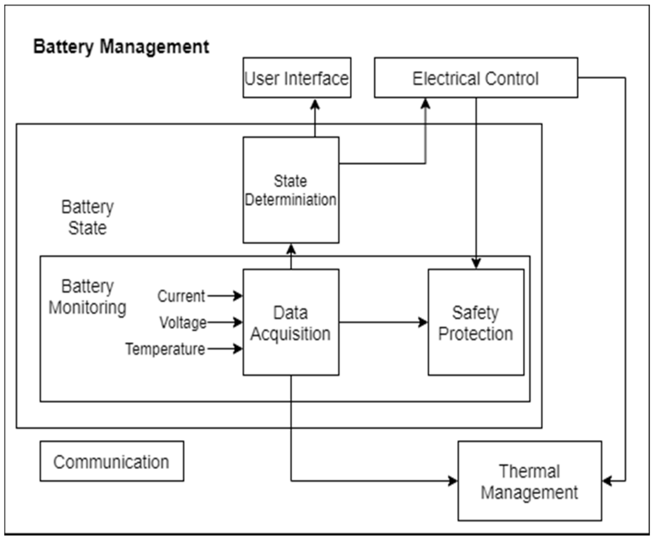


Fig 1. Block Diagram BMS

B. Need for Battery Management System

Li-ion batteries pack have a huge size and they are highly instable in nature. Rechargeable batteries should neither be over-charged nor under discharged at any condition which implies that there is a need to monitor and maintain its voltage and current. Monitoring is tough because there are a lot of cells fitted together either in series or parallel to make a battery pack in Electric Vehicle and each cell must be individually monitored for its safety and efficient operation which requires a specially dedicated system called the Battery Management System. Figure 2 shows the primary functions performed by BMS.

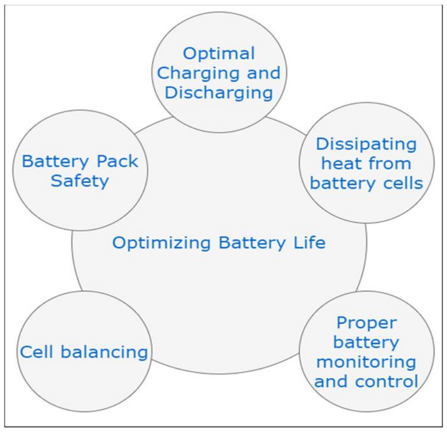

Optimal Charging and Discharging
Dissipating heat from battery cells
Proper battery monitoring and control
Cell balancing
Battery Pack Safety
Optimizing Battery Life

Fig 2: Functions of BMS

C. Need for Li-ion Battery

Lithium-ion batteries (LIBs) of different types offer potential benefits in terms of power density, cost, protection, efficiency and versatility in design.

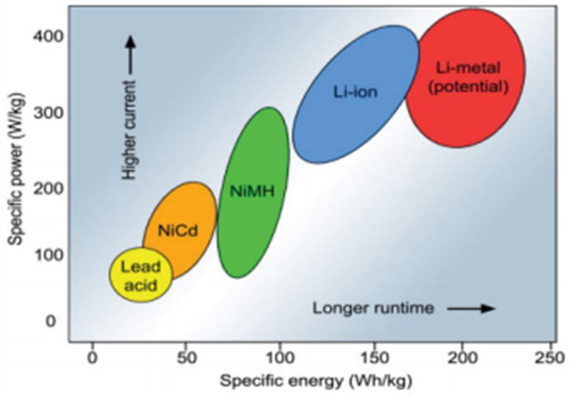


Fig 3: Specific energy and specific power of rechargeable batteries

Table 1: Various chemistries and their characteristics

	NiCd	NiMH	Lithium-ion	Lithium-ion polymer
Operating voltage	1.2 V	1.2 V	3.7 V	3.7 V
Energy density (Wh/kg)	60	80	>100	>100
High current performance	Good	Good	Satisfactory	Good
Cycle life	<500	<300	>500	>1000
Self-discharge	5	15-30	<5	<5
Specific advantage	Low cost	Higher energy	Higher energy	Higher energy and power density

D. SOC

The ratio of the available capacity to the rated capacity is defined as SOC. The equation 1 below gives the generalised SOC for LIBs.

$$SOC(t) = SOC(t-1) + \int_0^t \frac{I}{C_{bat}} \cdot dt \qquad (1)$$

where,

SOC(t): Battery state of charge at time ‘t’

SOC(t-1): Battery initial state of charge

I: Charge/Discharge current

t: Time(hr.)

The charging current consumed by the battery changes around the same operating conditions when charging a certain battery, as per the battery's SOC.

The absorbed charging current is substantially higher when the battery has a low SOC but the consumed charging current is considerably lower when the battery tends to have a high SOC.

II. LEVELS OF TESTING

A. Product Level Testing

Tests are categorized based on their SDLC (Software Development Life Cycle) status or the level of information they provide. There are usually four different levels, namely: unit testing, system testing, integration testing and acceptance testing.

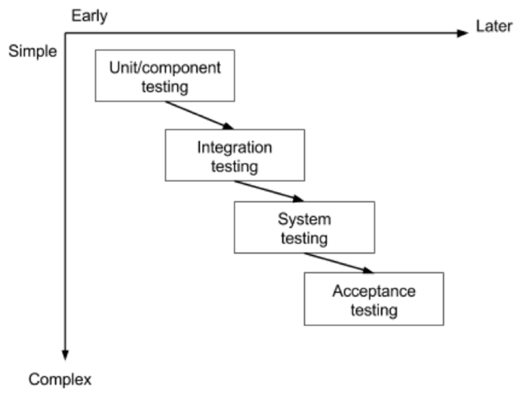


Fig 4: Levels of Testing

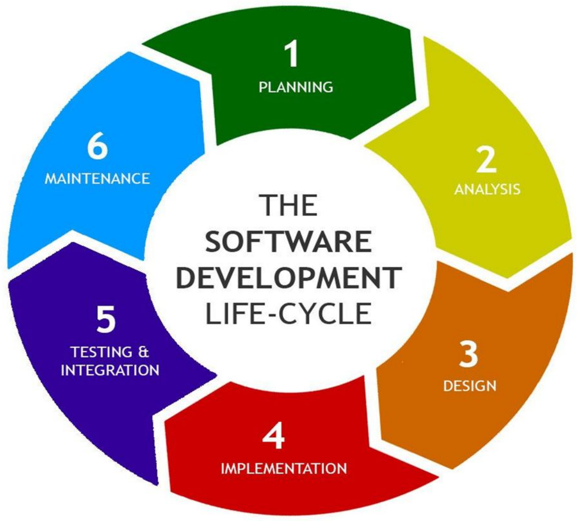


Fig 5: Software Development Cycle

There are mainly four testing levels which are:

1) *Unit Testing:* A Unit is the smallest testable device or program component that can be compiled, valued, assembled, and executed. This is carried out by developers.
2) *Integration Testing:* In a group, the combination of various software modules and evaluation is done to ensure that the embedded system is ready for the next test, i.e. system test.
3) *System Testing:* It is done on a comprehensive, integrated system. It allows users to monitor the conformity of a system according to customer requirement and tests the overall component interaction.
4) *Acceptance Testing:* It is conducted to determine if a configuration or contract conditions have been met.

B. Automation Testing Levels

Automation testing starts from Level 0 (No Autonomy), all the way up to Level 5 (Full Automation) as shown in figure 6.

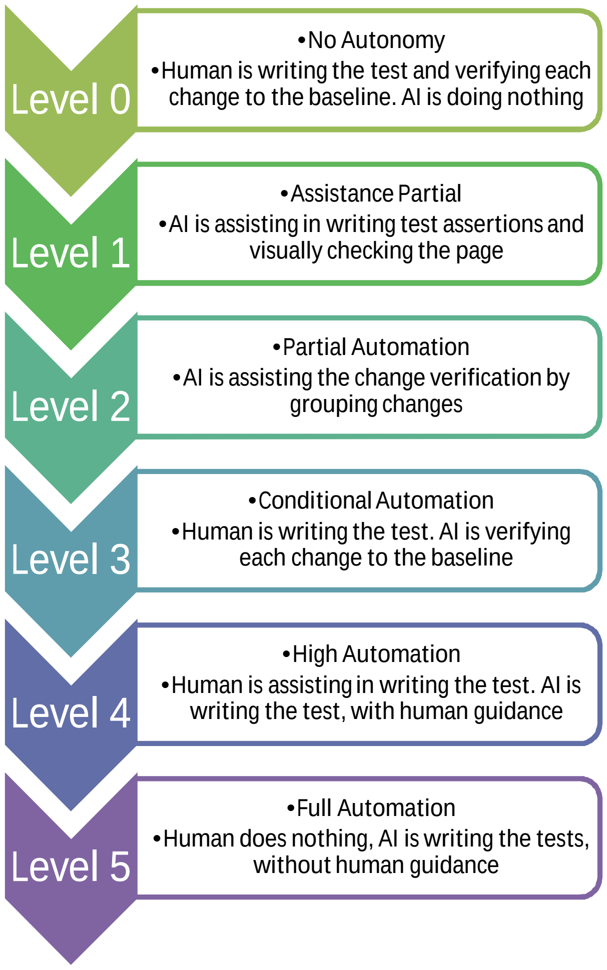

Level 0
•No Autonomy
•Human is writing the test and verifying each change to the baseline. AI is doing nothing
Level 1
•Assistance Partial
•AI is assisting in writing test assertions and visually checking the page
Level 2
•Partial Automation
•AI is assisting the change verification by grouping changes
Level 3
•Conditional Automation
•Human is writing the test. AI is verifying each change to the baseline
Level 4
•High Automation
•Human is assisting in writing the test. AI is writing the test, with human guidance
Level 5
•Full Automation
•Human does nothing, AI is writing the tests, without human guidance

Fig 6: Automation Testing Levels

C. Other Types of Testing

1) *Regression Testing:* It is a software testing used to confirm that a new change in program or code has not adversely affected existing features.
2) *Buddy Testing:* It is a methodology in which team members are working on the very same software program and on the same device, one person is working with the systems (with the mouse and the keyboard), and the other is making notes and instances.
3) *Alpha Testing:* It is a sort of acceptance test, conducted before publishing the product to the customers, to find all potential problems, bugs or major issues.
4) *Beta Testing:* Beta Testing of an application is carried out in a "real environment" by consumers of the software application and can be viewed as a form of external user acceptance testing.

III. NLP AND ITS FEATURES

A. Natural Language Processing

NLP is an Artificial Intelligence domain that provides the computers with the ability to interpret, comprehend and infer meaning from human languages, it refers to the interaction between humans and computers using the natural language. NLP techniques used to decode human languages are based on ML.

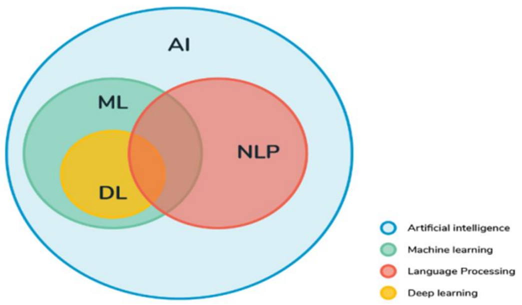


Fig 6: Correlation of AI, ML, NLP and Deep Learning

B. Tokenization

The method of breaking text into words and sentences called tokens and expelling some characters, such as punctuation, simultaneously.

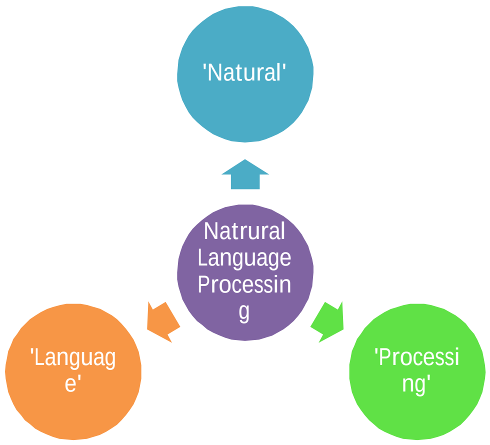


Fig 7: Example of Tokenisation

C. Filtered Sentence

It is getting rid of common articles, pronouns and prepositions in the English language like "and," "the" or "to" etc. In this methodology, very common words that give no or little value to the NLP goal are filtered and discarded from the text to be analyzed, removing therefore frequent terms that are not informative about the given text. There is no standardized list of words with stops.

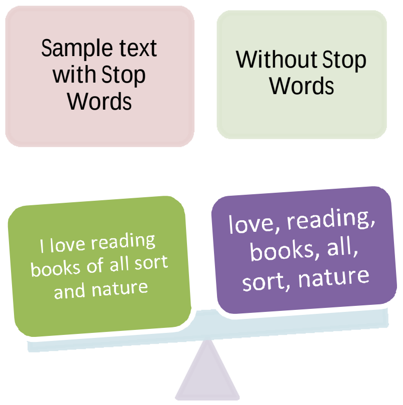

Sample text with Stop Words
Without Stop Words
I love reading books of all sort and nature
love, reading, books, all, sort, nature

Fig 8: Example of Filtered Sentence

D. POS Tagging
POS tagging is the technique of marking a word in a corpus, based on its context and definition, into a corresponding part of a speech tag.

Table 2: List of commonly used POS-tags

FW	Foreign Word
IN	Preposition/Subordinating Conjunction
JJ	Adjective
NN	Noun, singular or mass
NNP	Proper Noun, singular
PP	Prepositional Phrase
VP	Verb Phrase
RB	Adverb
RP	Particle
SYM	Symbol
VB	Verb, the base form
UH	Interjection

E. Stemming
Refers to a process of cutting the end or start of words with the aim to remove affixes (lexical additions to the word root). It may be used to correct tokens spelling errors. Stemmers are very quick to use and to run (they do basic actions on a string).

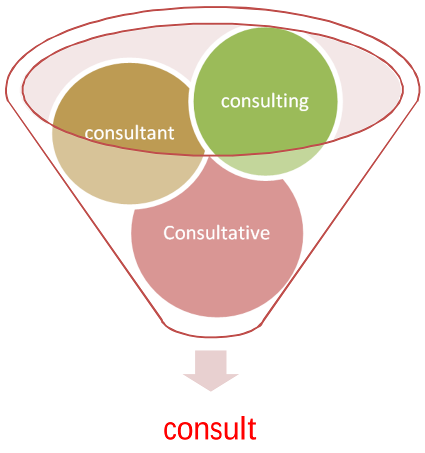


Fig 10: Example of Stemming

F. Bag of Words
Model that counts all the words in a piece of text and creates a matrix for the sentence or document, irrelevant of grammar or word order. These word frequencies or occurrences are then used as features for training a classifier.

G. Lemmatization
It is an approach of reducing a word to its base form, and afterwards grouping together all the various forms of the same word. For example, verbs in the past tense are modified into the present (e.g. "went" is modified to "go") and synonyms are merged (e.g. "best" is changed to "good"), essentially standardizing any word into a meaning close to its root.
Lemmatization can distinguish between the same words which have different meanings depending on the particular context.

H. Reg-ex

Regular expressions are an effective tool for manipulating strings. All modern languages have library packages similar to those for regular expressions. Regular expressions are used for:

1) Searching a string (search and match)
2) Replacing parts of a string or substring.
3) Break strings into smaller pieces or splitting.

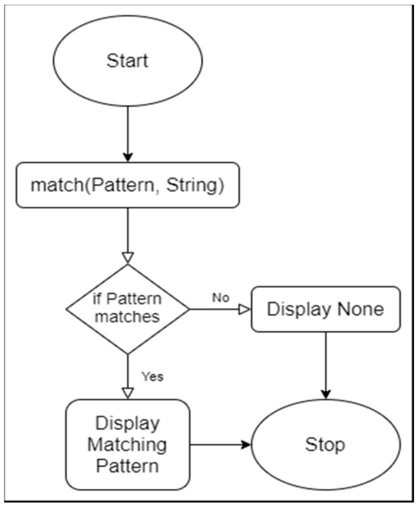


Fig 11: Flowchart of Reg-ex

Table 3: Commonly used reg-ex patterns

^	Matches beginning of line
$	Matches the end of line
.	Matches any single character except newline
[…]	Matches any single character in brackets
\w	Matches word characters
\s	Matches whitespace
\d	Matches digits
\A	Matches beginning of the string
[a-z]	Matches any lowercase ASCII letter
[^aeiou]	Matches anything other than a lowercase vowel

IV. METHODOLOGY

A. Coulomb Counting Method for SOC Estimation

Coulomb counting method measures the discharging current of a battery and integrates the discharging current over time to estimate *SOC(t).* The equation 2 gives the coulomb counting method in LIBs.

$$SOC(t) = SOC(t-1) + \frac{I(t)}{Q_n}\Delta t \qquad (2)$$

Factors reducing the estimation accuracy of coulomb counting are charge and discharge efficiencies, temperature, and cycle life.
For model-based SOC estimation and BMS improvement, among different models, equivalent circuit models are used. Simplicity and relatively few numbers of parameters are the main characteristics for designing these models. An equivalent circuit model is composed of basic elements, resistors, capacitors, and a voltage source, in the form of a circuit network.
Thus, the first-order model is discussed below to understand the real-time applications, that is commonly used in terms of accuracy and sophistication for its good characteristics. The battery model used here is a voltage source designed to represent the open-circuit

voltage, an ohmic resistance 'R_i' modelling the internal resistance and a single polarization RC network describing the polarization phenomena.
This model is given below, where I_{bat} is the battery charge or discharge current and V_{bat} is the battery terminal voltage.

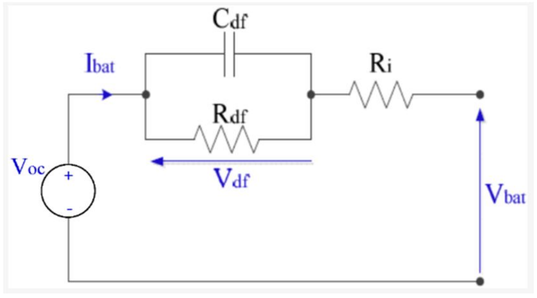
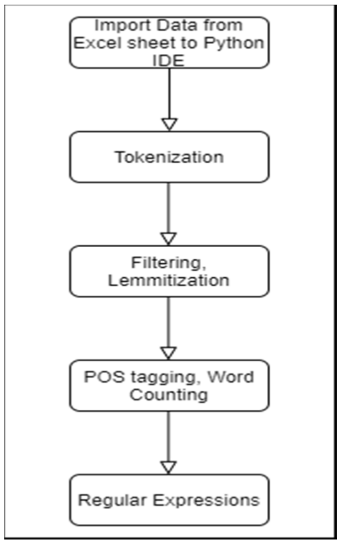


Fig 12: Equivalent circuit model

B. Automation of Test Specification sheet

First, the test specification sheet is imported into the python editor, generally Anaconda Spyder / Visual Studio Code. Then the Excel sheet is extracted using Pandas library. After that, various Natural Language Processing functions are applied to get the relevant instruction from the cells. They are matched across a predefined Corpus. Corpus can be modified according to the OEM requirements. Regex is generally used for comparison-based model and an output is generated which gives the relevance of the efficiency with which the word is predicted to be correlated to the corpus. The flowchart of the whole process is given in the figure 13 below.


Import Data from Excel sheet to Python IDE
Tokenization
Filtering, Lemmitization
POS tagging, Word Counting
Regular Expressions
Fig 13: Flowchart of Methodology

C. Results And Discussion

1) An easy build-up of testing specification sheet by using python scripting and reduction in manual errors generated and compiled in the final documentation.
2) Use of Regular expressions package from python has greatly increased the ease with which the test specifications can be clustered and modified.
3) Usage of POS-tagging, Bag of Words, Vectorization, Tokenization, Regex, Counter has greatly helped in improving the NLP using python and automation of manual tasks.

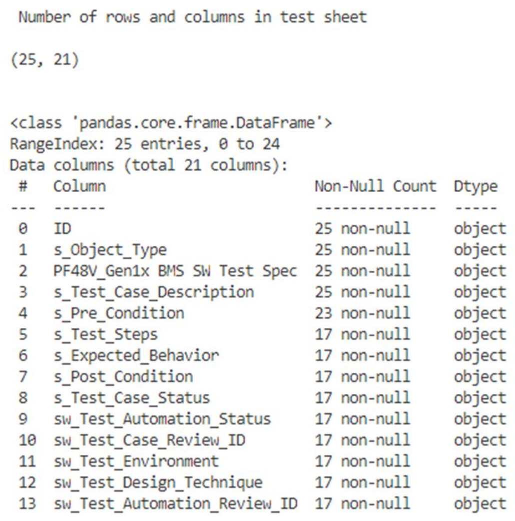
Fig 14 shows the output of the general information extracted from the test specification sheet, which would be later used to formulate algorithms and decide the flow of the NLP. First the test specification sheet is loaded in the compiler and then the code is run to extract the rows and columns and their contents in a sequential manner.
Fig 15 shows the contents of the data frame created by taking the contents of a single column of the test specification sheet and it is printed in the compiler output window, this gives an idea about further steps to be implemented to remove the unnecessary contents and further refining the corpus.

Fig 16 represents the list of tokenized words implemented on the data frame and thus, each word becomes a token. Tokenisation helps in filtering the data to apply further processing.

Fig 17 shows the stem word of each word or the root word of each tokenized value.

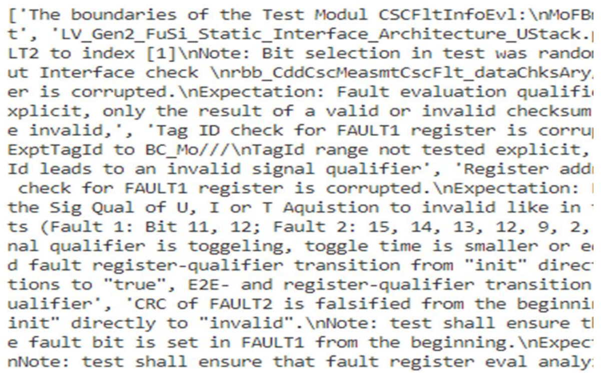
Fig 18 shows the lemma form of each tokenized word in the data frame whereby the filtering and different NLP techniques is applied.

Fig 19 shows the visual representation of the frequency of top ten words in the data frame of tokenized words from the test specification sheet.



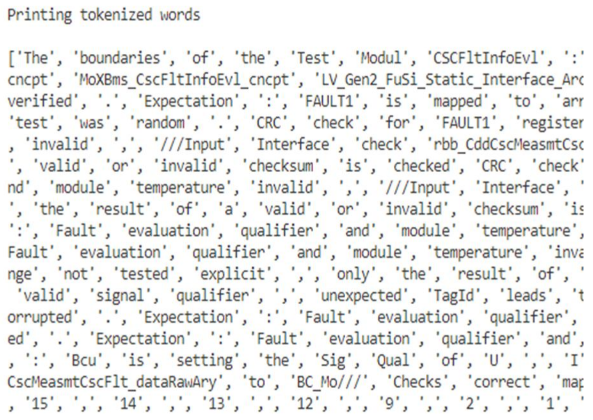
```
 Number of rows and columns in test sheet

(25, 21)


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 21 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   ID                         25 non-null     object
 1   s_Object_Type              25 non-null     object
 2   PF48V_Gen1x BMS SW Test Spec  25 non-null   object
 3   s_Test_Case_Description    25 non-null     object
 4   s_Pre_Condition            23 non-null     object
 5   s_Test_Steps               17 non-null     object
 6   s_Expected_Behavior        17 non-null     object
 7   s_Post_Condition           17 non-null     object
 8   s_Test_Case_Status         17 non-null     object
 9   sw_Test_Automation_Status  17 non-null     object
 10  sw_Test_Case_Review_ID     17 non-null     object
 11  sw_Test_Environment        17 non-null     object
 12  sw_Test_Design_Technique   17 non-null     object
 13  sw_Test_Automation_Review_ID 17 non-null   object
```

Fig 14: Number of rows and columns and their general information



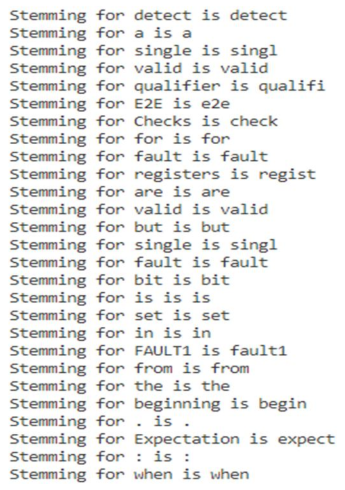
```
['The boundaries of the Test Modul CSCFltInfoEvl:\nMoFB
t', 'LV_Gen2_FuSi_Static_Interface_Architecture_UStack.
LT2 to index [1]\nNote: Bit selection in test was rando
ut Interface check \nrbb_CddCscMeasmtCscFlt_dataChksAry.
er is corrupted.\nExpectation: Fault evaluation qualifi
xplicit, only the result of a valid or invalid checksum
e invalid,', 'Tag ID check for FAULT1 register is corru
ExptTagId to BC_Mo///\nTagId range not tested explicit,
Id leads to an invalid signal qualifier', 'Register add
 check for FAULT1 register is corrupted.\nExpectation:
the Sig Qual of U, I or T Aquistion to invalid like in
ts (Fault 1: Bit 11, 12; Fault 2: 15, 14, 13, 12, 9, 2,
nal qualifier is toggeling, toggle time is smaller or e
d fault register-qualifier transition from "init" direc
tions to "true", E2E- and register-qualifier transition
ualifier', 'CRC of FAULT2 is falsified from the beginni
init" directly to "invalid".\nNote: test shall ensure t
e fault bit is set in FAULT1 from the beginning.\nExpec
nNote: test shall ensure that fault register eval analy
```

Fig 15: Data frame of one column



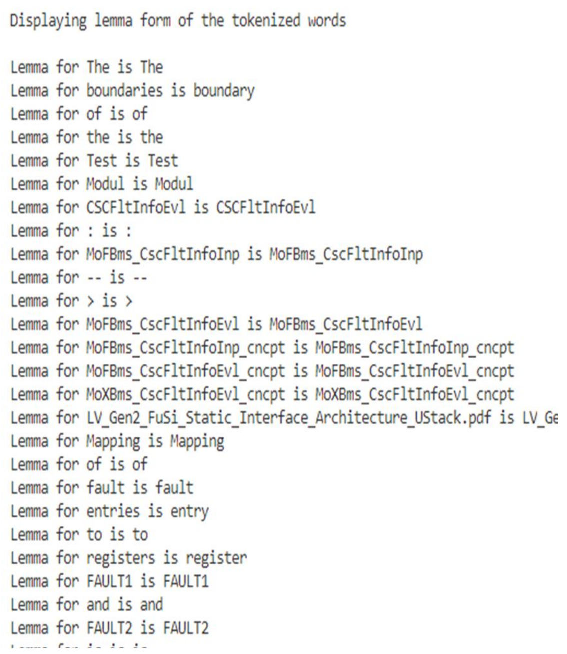
```
Printing tokenized words

['The', 'boundaries', 'of', 'the', 'Test', 'Modul', 'CSCFltInfoEvl', ':'
cncpt', 'MoXBms_CscFltInfoEvl_cncpt', 'LV_Gen2_FuSi_Static_Interface_Ar
verified', '.', 'Expectation', ':', 'FAULT1', 'is', 'mapped', 'to', 'ar
'test', 'was', 'random', '.', 'CRC', 'check', 'for', 'FAULT1', 'register
, 'invalid', ',', '///Input', 'Interface', 'check', 'rbb_CddCscMeasmtCs
', 'valid', 'or', 'invalid', 'checksum', 'is', 'checked', 'CRC', 'check'
nd', 'module', 'temperature', 'invalid', ',', '///Input', 'Interface', '
', 'the', 'result', 'of', 'a', 'valid', 'or', 'invalid', 'checksum', 'is
':', 'Fault', 'evaluation', 'qualifier', 'and', 'module', 'temperature',
Fault', 'evaluation', 'qualifier', 'and', 'module', 'temperature', 'inva
nge', 'not', 'tested', 'explicit', ',', 'only', 'the', 'result', 'of', '
 'valid', 'signal', 'qualifier', ',', 'unexpected', 'TagId', 'leads', 't
orrupted', '.', 'Expectation', ':', 'Fault', 'evaluation', 'qualifier',
ed', '.', 'Expectation', ':', 'Fault', 'evaluation', 'qualifier', 'and',
, ':', 'Bcu', 'is', 'setting', 'the', 'Sig', 'Qual', 'of', 'U', ',', 'I'
CscMeasmtCscFlt_dataRawAry', 'to', 'BC_Mo///', 'Checks', 'correct', 'ma
, '15', ',', '14', ',', '13', ',', '12', ',', '9', ',', '2', ',', '1', '
```

Fig 16: Tokenized words of the data frame



```
Stemming for detect is detect
Stemming for a is a
Stemming for single is singl
Stemming for valid is valid
Stemming for qualifier is qualifi
Stemming for E2E is e2e
Stemming for Checks is check
Stemming for for is for
Stemming for fault is fault
Stemming for registers is regist
Stemming for are is are
Stemming for valid is valid
Stemming for but is but
Stemming for single is singl
Stemming for fault is fault
Stemming for bit is bit
Stemming for is is is
Stemming for set is set
Stemming for in is in
Stemming for FAULT1 is fault1
Stemming for from is from
Stemming for the is the
Stemming for beginning is begin
Stemming for . is .
Stemming for Expectation is expect
Stemming for : is :
Stemming for when is when
```

Fig 17: Root form of the tokenized words



```
Displaying lemma form of the tokenized words

Lemma for The is The
Lemma for boundaries is boundary
Lemma for of is of
Lemma for the is the
Lemma for Test is Test
Lemma for Modul is Modul
Lemma for CSCFltInfoEvl is CSCFltInfoEvl
Lemma for : is :
Lemma for MoFBms_CscFltInfoInp is MoFBms_CscFltInfoInp
Lemma for -- is --
Lemma for > is >
Lemma for MoFBms_CscFltInfoEvl is MoFBms_CscFltInfoEvl
Lemma for MoFBms_CscFltInfoInp_cncpt is MoFBms_CscFltInfoInp_cncpt
Lemma for MoFBms_CscFltInfoEvl_cncpt is MoFBms_CscFltInfoEvl_cncpt
Lemma for MoXBms_CscFltInfoEvl_cncpt is MoXBms_CscFltInfoEvl_cncpt
Lemma for LV_Gen2_FuSi_Static_Interface_Architecture_UStack.pdf is LV_Ge
Lemma for Mapping is Mapping
Lemma for of is of
Lemma for fault is fault
Lemma for entries is entry
Lemma for to is to
Lemma for registers is register
Lemma for FAULT1 is FAULT1
Lemma for and is and
Lemma for FAULT2 is FAULT2
```

Fig 18: Lemma form of the tokenized data frame

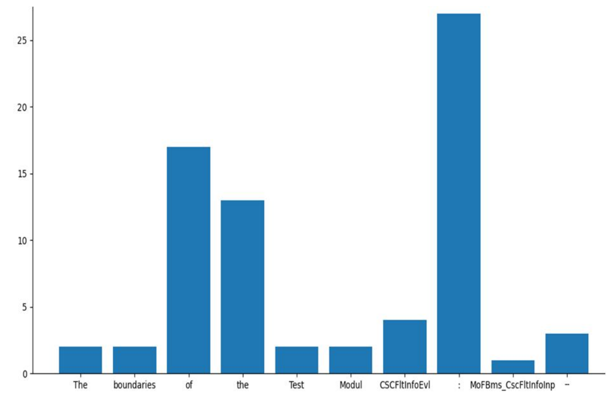


Fig 19: Frequency of top ten tokenized words from the data frame

V. CONCLUSION AND FUTURE SCOPE

Due to increases in electric automobiles on the road, the risk of fatal accidents is increasing day by day. Hence battery testing plays a major role before the deployment for electric cars production. Earlier testing is carried out manually. Accuracy of the manual process is very less and also, redundant in time aspects.

In this research, the model was created in MATLAB-SIMULINK software for estimating the SOC values which will give the amount of capacity or charge remaining in the LIB. The state-of-charge (SOC) was estimated successfully and its value decreases with the time was obtained in graphical form by sing the Coulomb Counting Method. The battery SOC obtained using Coulomb counting method depends on battery current measurement.

A. Conclusion

This research paper is carried out to provide efficiency in time and cost over the currently used method. Humans are prone to make errors and thus automating the task of creating a corpus will lead to a reduction in human errors and gross errors. Reduction in the boring task done by humans and thus, the time saved can be used to complete more challenging tasks.
Automation of the corpus creation at software level battery testing was successful, which resulted in the reduction of time by almost 70-75% compared to the existing process. Implementing Automation integrates robustness into the existing system and ensures scalability, availability and reliability.

B. Future Scope

If any error occurs a person has to check it and need to rectify the error, so if machine learning and artificial intelligence are used instead, it can handle the errors by its own. The system will make its own decision if there is an error in the corpus or test specification sheet which will eliminate human intervention. It is particularly beneficial while dealing with data of high scale which is usually the case.

REFERENCES

[1] Mr. Javed H. Shaikh, “Battery management system in electric vehicle”, 7th International Conference on Recent Trends in Engineering, Science & Management, 2017.
[2] Chao Yu, “Cost-efficient Thermal Management for a 48V Li-ion Battery in a Mild Hybrid Electric Vehicle”, China Society of Automotive Engineers (China SAE), 2018.
[3] Fabrizio Pastore, Arda Goknil, Lionel C. Briand, “Automatic Generation of System Test Cases from Use Case Specifications: an NLP-based Approach”, arxiv.org, 2019
[4] Vishal Sangave, “Generic Test Automation”, International Journal of Science and Research (IJSR), 2015
[5] Mykhailo Lobur, Andriy Romanyuk, Mariana Romanyshyn, “Using NLTK for educational and scientific purposes”, 11th International Conference the Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), 2014
[6] Andres Arellano, Edward Carney, “Natural Language Processing of Textual Requirements”, 10th International Conference on Systems, 2015
[7] Truong M. N. Bui, Mona Faraji Niri, Daniel Worwood, “An advanced Hardware-in-the-Loop Battery Simulation Platform for the Experimental Testing of Battery Management System”, 23rd International Conference on Mechatronics Technology (ICMT), 2019
[8] Budhy Rahmawatie, Wahyudi Sutopo, Agus Purwanto, “Designing framework for standardization and testing reqirements of battery management system for electric vehicle application”, 4th International Conference on Electric Vehicular Technology (ICEVT), 2017
[9] Maitane Berecibar, Maitane Garmendia, “State of health estimation algorithm of $LiFePO_4$ battery packs based on differential voltage curves for battery management system application”, Energy Journal Elsevier, 2016
[10] Anton E Pavlov, Dmitry V Telyshev, “Calibration Module for Battery Management System of Medical Devices”, IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 2019
[11] Yuan Liu, Xin Qiao, Zhi-Xue Wang, “The monitoring and fault diagnosis technology research of battery management system”, Chinese Automation Congress (CAC), 2017
[12] Chaoran Li, Yaxiang Fan, Guorun Yang, “A Recurrent Neural Network with Long Short-Term Memory for State of Charge Estimation of Lithium- ion Batteries”, IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), 2019
[13] Yuchen Song, Lyu Li, Yu Peng, “Lithium-Ion Battery Remaining Useful Life Prediction Based on GRU-RNN”, 12th International Conference on Reliability, Maintainability, and Safety (ICRMS), 2018
[14] Al Sweigart, Automate the Boring Stuff with Python, 14th April 2015
[15] Steven Bird, Ewan Klein, and Edward Loper, Python Natural Language Processing, June 2009
[16] Victor Romero, Regular Expressions in Python, Packt Publishing Ltd, 2014
[17] A. F. Burke, “Batteries and ultracapacitors for electric, hybrid, and fuel cell vehicles,” Proc. IEEE, vol. 95, no. 4, pp. 806–820, Apr. 2007
[18] Sun, K., Shu, Q, “Overview of the types of battery models”, IEEE 30th Chinese Control Conference (CCC), China, July 2011.
[19] R.J. Chen, H.Y. Lin, “Electric circuit modelling for lithium-ion batteries by intermittent discharging”, IET Power Electron, vol. 7 no. 10, October 2014.