

Automatic Text Summarization using NLP

Ms.M.ANITHA¹, Ms.K.PAVANI², Ms.G.MAHATHI SAI PRIYA³

#1 Assistant professor in the Department of Master of Computer Applications in the SRK Institute of Technology, Enikepadu, Vijayawada, NTR District

#2 Assistant professor in the Department of Master of Computer Applications SRK Institute of Technology, Enikepadu, Vijayawada, NTR District

#3 MCA student in the Department of Master of Computer Applications at SRK Institute of Technology, Enikepadu, Vijayawada, NTR District

ABSTRACT_ The application of "artificial intelligence" in the field of "natural language processing" (NLP) enables computers and other computerized systems to understand general commands or text in the same format in which humans interact. Natural language processing is a sub field of "artificial intelligence." In this process, "machine learning" is used to offer accuracy and pattern of the identification process in order to increase the efficiency of the interaction process. This is done to improve the user experience.

The summaries generated by these models were put to the test by being compared to a variety of datasets, including the CNN corpus, DUC2000, single and multiple text documents, and others. Long articles are common on websites that cover news, blogs, and consumer reviews, among other types of websites. We have investigated not only these methodologies, but also their tendencies, successes, previous work, and potential future scope in text summarizing, among other areas of study. The majority of the techniques discussed in this article provide either abstractive or extractive summaries of the text documents they are applied to. The abundance of data that is available nowadays has made it very necessary to summarize lengthy texts in order to get the exact quantity of information that is needed from those writings. The majority of the discussion in this study is centered on the structured and semantic techniques to summarizing the text documents that were used. Research has been conducted on a number of articles to investigate the several approaches to text summary that have been tried up to this point.



1.INTRODUCTION

The world of today is based on computers and data. Our intangible ideas and imaginations are data. At the same time, we are data consumers and producers. Data can come from or come from anything in our everyday lives. For instance, when we drive, data like the speed of the car, mileage, distance traveled, and so on are involved. Data has been a significant part of our lives since the 20th century, but today we infer more from data. We store them and use wireless and electronic systems to get to them. There has been an enormous amount of data available since the internet's inception. Data is stored on the Internet. News, movies, education, health, medicine, countries, weather, geology, and other topics are covered. is readily available online. Data that is statistical, numerical, mathematical, or textual could be used. The greater number of characters in text data makes it more challenging to interpret it. There must be a system that allows us to access only the essential portions of this enormous amount of information. One approach is to summarize text. Since decades ago, text summarization has been the subject of research and study. Concise summaries have been produced by proposing and

evaluating various models on various datasets. Different comparison scores are used to compare them. EXT or ABS, single or multiple documents, query-based or generic, and text summarization are all options. EXT text summarization is a method for producing summaries that adhere to the document's sentences. The ABS is more general and focuses on the document's key ideas. Likewise, single report synopsis strategies give rundowns of the text of a solitary record, and multidocument produces outlines of various records. Additionally, it is necessary to summarize text based on queries these days. Generic summaries are mostly ABS that focus on the general area of the text input, whereas query-based summarization models provide summaries of the text based on a specific area as described by the user's query. Text summarization has been widely used in a variety of fields, including engineering, law, science, and medicine, among others. Specialists have zeroed in on creating synopses of specialist's solutions, and that has been demonstrated extremely helpful to patients. In a similar vein, lengthy news articles have been summarized so that readers can learn a lot about a variety of subjects in a short amount of time[1]. For the past five years, we have discussed the



various text summarization methods in this paper. The most widely recognized techniques were viewed as Machine Learning (ML), NNs, support learning, arrangement to succession demonstrating and fluffy rationale. The proposed objective function has also been optimized using a variety of optimization techniques for the purpose of text summarization. We can see that different methods were tested on the same dataset and found to have different accuracy scores. In addition, we can see that some researchers have discovered that the summaries produced by combining the various approaches are more accurate than those produced by employing only one. Python libraries like scikit learn, nltk, spacy, and fastai have been used when NLP processing has been used to summarize text documents.

2. LITERATURE SURVEY

2.1 C. Ordonez, Y. Zhang and S. L. Johnsson, "Scalable machine learning computing a data summarization matrix with a parallel array DBMS", Distrib. Parallel Databases, vol. 37, no. 3, pp. 329-350, 2019

Machine learning models in big data analytics must be processed in a way that is scalable (above RAM limits) and highly parallel (using many CPU cores), both of

which typically involve extensive matrix manipulation. A promising method for manipulating large matrices are array DBMSs. In light of this, we present a fast system that makes use of a parallel array database management system (DBMS) to evaluate a comprehensive yet compact matrix summarization that is beneficial to numerous machine learning models. We concentrate on two typical models: PCA (unsupervised) and linear regression (supervised). Our methodology consolidates information synopsis inside the equal DBMS with additional model calculation in a numerical language (for example R). On one node, we present a two-phase algorithm that evaluates matrix equations with reduced intermediate matrices in main memory after first computing a general data summary in parallel. Speedup and time/space complexity are characterized by the theories we present here. In a shared-nothing architecture, we take scale-up and scale-out into account from a parallel data system perspective. Our system is based on array operators written in C++ and operates directly on the Unix file system as opposed to Java or Scala running on HDFS mounted on top of Unix, which results in much faster processing than the majority of big data analytics systems. Our system

outperforms Spark (parallel) and R (single machine) by orders of magnitude in terms of time efficiency in experiments. Parallel benchmarks with varying processing nodes and thread counts are presented by us. Analysts should be encouraged to use a parallel array DBMS for matrix summarization by our two-phase strategy.

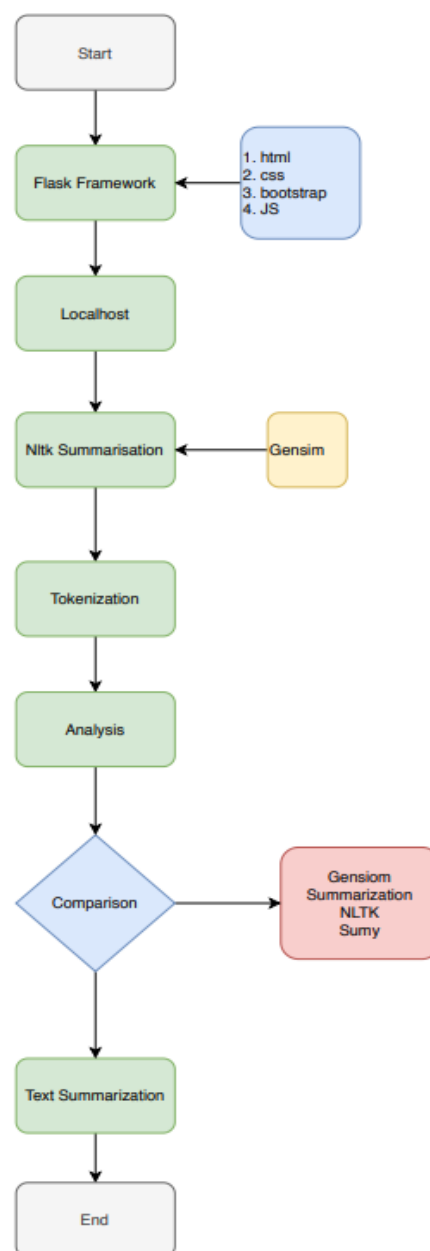
2.2 G. Silva, R. Ferreira, S. J. Simske, L. Rafael Lins, M. Riss and H.O. Cabral, "Automatic text document summarization based on machine learning", DocEng 2015 - Proc. 2015 ACM Symp. Doc. Eng, pp. 191-194, 2015.

The requirement for programmed age of outlines acquired significance with the phenomenal volume of data accessible in the Web. To produce a summary, extractive summarization-based automated systems select the most significant sentences from one or more texts. This article integrates twenty of the most frequently cited extractive summarization strategies into a tool to evaluate their quality using Machine Learning methods. In this evaluation, both quantitative and qualitative factors were taken into account to demonstrate the scheme's validity. The CNN-corpus, possibly the largest and most suitable test corpus currently available for

evaluating extractive summarization strategies, served as the subject of the experiments.

3.PROPOSED WORK

We have also seen many researchers use



deep learning techniques for EXT as well



as ABS text summarization. Deep learning is an area of ML. Various NN techniques have been used. Similarly, reinforcement learning, Convolutional NN(CNN), RNN have also been applied to generate text summaries[10]. There's also a study of sequence-to-sequence models for text

1:Architecture

summarization these days. These methods are extension of ML. We describe some of the papers that use the above mentioned methods to generate summaries of text. \

3.1 IMPLEMENTATION

To implement NLP-based machine learning approaches for text summarization, you will need to use various Python libraries and modules. Here are some commonly used modules that you may need:

1. Natural Language Toolkit (NLTK): NLTK is a popular Python library for NLP tasks, including text preprocessing, tokenization, and part-of-speech tagging.
2. Gensim: Gensim is a library for unsupervised topic modeling and natural language processing. It provides tools for building topic models, similarity queries, and summarization.

3. Scikit-learn: Scikit-learn is a popular machine learning library for Python that includes tools for classification, regression, clustering, and dimensionality reduction.

4. PyTorch: PyTorch is an open-source machine learning framework that provides tools for building and training neural networks, including transformers that can be used for text summarization.

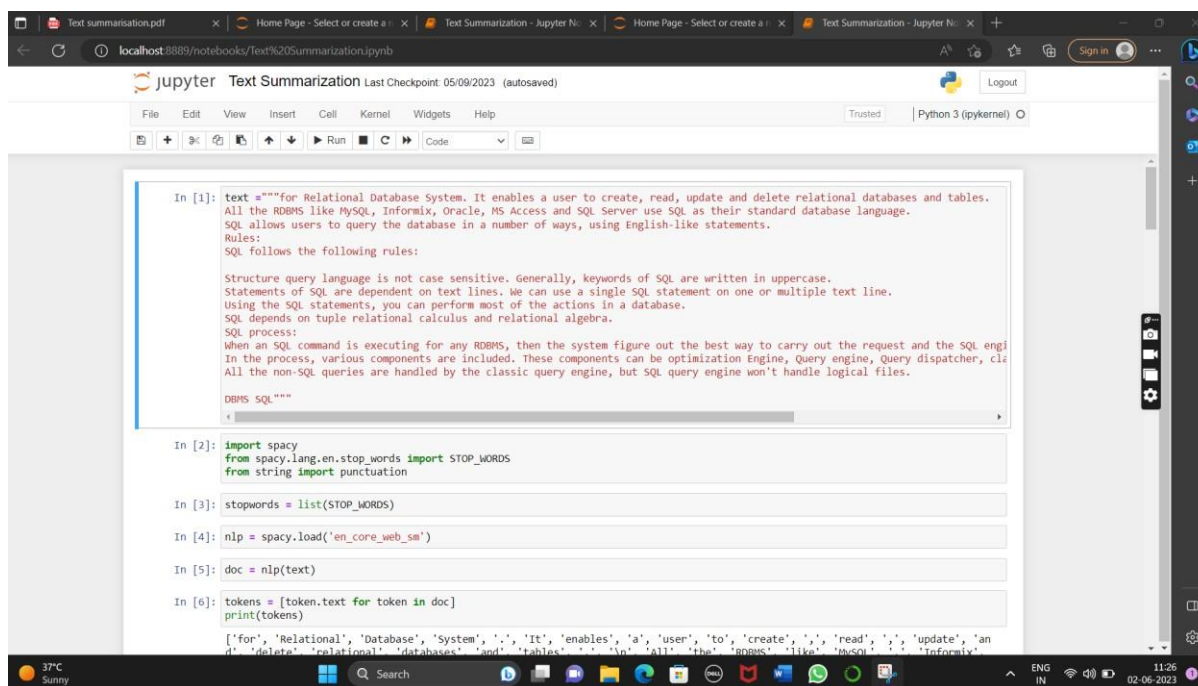
5. TensorFlow: TensorFlow is another popular machine learning framework that can be used for building and training neural networks, including those used for text summarization.

6. SpaCy: SpaCy is a library for advanced NLP tasks, including named entity recognition, dependency parsing, and text classification.

7. Sumy: Sumy is a library for automatic text summarization. It provides several summarization algorithms, including TextRank and LSA, that can be used for extractive summarization.

8. Rouge: Rouge is a Python package for evaluating automatic summarization systems. It provides metrics for comparing summaries generated by machine learning models to human-generated summaries

4.RESULTS AND DISCUSSION



```

In [1]: text = """for Relational Database System. It enables a user to create, read, update and delete relational databases and tables.
All the RDBMS like MySQL, Informix, Oracle, MS Access and SQL Server use SQL as their standard database language.
SQL allows users to query the database in a number of ways, using English-like statements.
Rules:
SQL follows the following rules:
Structure query language is not case sensitive. Generally, keywords of SQL are written in uppercase.
Statements of SQL are dependent on text lines. We can use a single SQL statement on one or multiple text line.
Using the SQL statements, you can perform most of the actions in a database.
SQL depends on tuple relational calculus and relational algebra.
SQL process:
When an SQL command is executing for any RDBMS, then the system figure out the best way to carry out the request and the SQL engine.
In the process, various components are included. These components can be optimization Engine, Query engine, Query dispatcher, etc.
All the non-SQL queries are handled by the classic query engine, but SQL query engine won't handle logical files.
DBMS SQL"""

In [2]: import spacy
from spacy.lang.en.stop_words import STOP_WORDS
from string import punctuation

In [3]: stopwords = list(STOP_WORDS)

In [4]: nlp = spacy.load('en_core_web_sm')

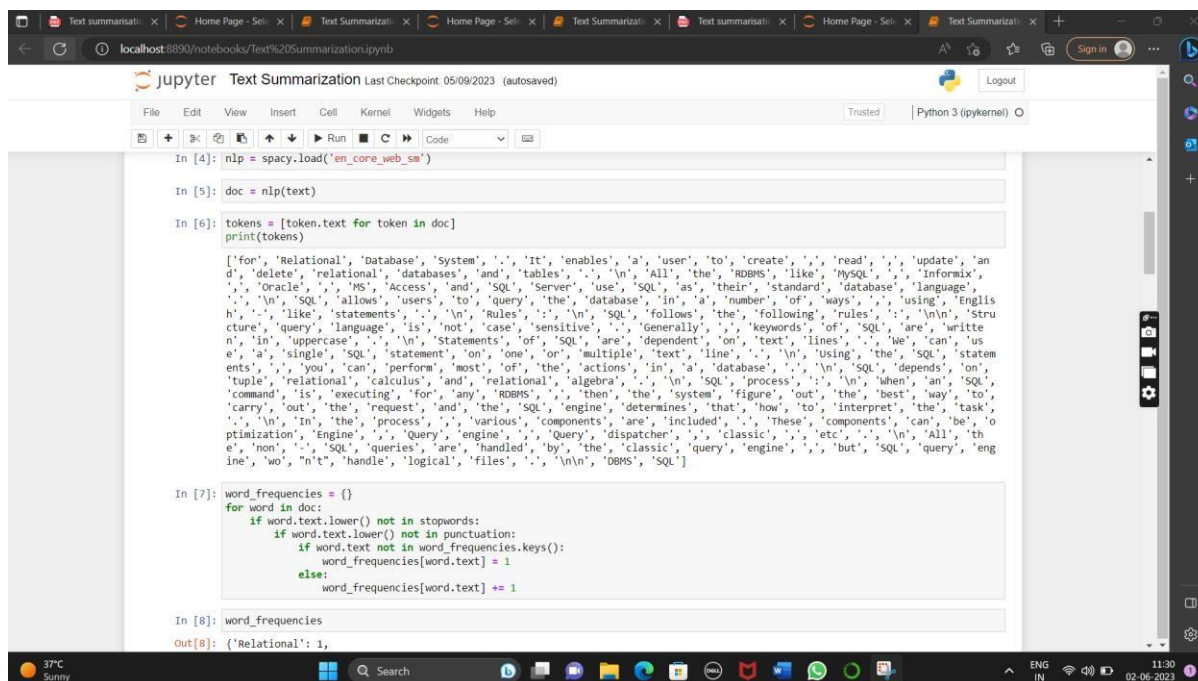
In [5]: doc = nlp(text)

In [6]: tokens = [token.text for token in doc]
print(tokens)

['for', 'Relational', 'Database', 'System', 'It', 'enables', 'a', 'user', 'to', 'create', 'read', 'update', 'an', 'delete', 'relational', 'databases', 'and', 'tables', 'All', 'the', 'RDBMS', 'like', 'MySQL', 'Informix', 'Oracle', 'MS', 'Access', 'and', 'SQL', 'Server', 'use', 'SQL', 'as', 'their', 'standard', 'database', 'language', 'SQL', 'allows', 'users', 'to', 'query', 'the', 'database', 'in', 'a', 'number', 'of', 'ways', 'using', 'English', 'like', 'statements', 'Rules', 'SQL', 'follows', 'the', 'following', 'rules', 'Structure', 'query', 'language', 'is', 'not', 'case', 'sensitive', 'Generally', 'keywords', 'of', 'SQL', 'are', 'written', 'in', 'uppercase', 'Statements', 'of', 'SQL', 'are', 'dependent', 'on', 'text', 'lines', 'We', 'can', 'use', 'a', 'single', 'SQL', 'statement', 'on', 'one', 'or', 'multiple', 'text', 'line', 'Using', 'the', 'SQL', 'statements', 'you', 'can', 'perform', 'most', 'of', 'the', 'actions', 'in', 'a', 'database', 'SQL', 'depends', 'on', 'tuple', 'relational', 'calculus', 'and', 'relational', 'algebra', 'SQL', 'process', 'When', 'an', 'SQL', 'command', 'is', 'executing', 'for', 'any', 'RDBMS', 'then', 'the', 'system', 'figure', 'out', 'the', 'best', 'way', 'to', 'carry', 'out', 'the', 'request', 'and', 'the', 'SQL', 'engine', 'determines', 'that', 'how', 'to', 'interpret', 'the', 'task', 'In', 'the', 'process', 'various', 'components', 'are', 'included', 'These', 'components', 'can', 'be', 'optimization', 'Engine', 'Query', 'dispatcher', 'etc', 'non', 'SQL', 'queries', 'are', 'handled', 'by', 'the', 'classic', 'query', 'engine', 'but', 'SQL', 'query', 'engine', 'won', 't', 'handle', 'logical', 'files', 'DBMS', 'SQL']

```

Screenshot 1 : packages installation for Grammar check and spell check The packages for the grammar check and spell check are included in the module for performing the grammar and spell corrections by using from gingerit.gingerit import GingerIt, !pip install gingerit.



```

In [4]: nlp = spacy.load('en_core_web_sm')

In [5]: doc = nlp(text)

In [6]: tokens = [token.text for token in doc]
print(tokens)

['for', 'Relational', 'Database', 'System', 'It', 'enables', 'a', 'user', 'to', 'create', 'read', 'update', 'an', 'delete', 'relational', 'databases', 'and', 'tables', 'All', 'the', 'RDBMS', 'like', 'MySQL', 'Informix', 'Oracle', 'MS', 'Access', 'and', 'SQL', 'Server', 'use', 'SQL', 'as', 'their', 'standard', 'database', 'language', 'SQL', 'allows', 'users', 'to', 'query', 'the', 'database', 'in', 'a', 'number', 'of', 'ways', 'using', 'English', 'like', 'statements', 'Rules', 'SQL', 'follows', 'the', 'following', 'rules', 'Structure', 'query', 'language', 'is', 'not', 'case', 'sensitive', 'Generally', 'keywords', 'of', 'SQL', 'are', 'written', 'in', 'uppercase', 'Statements', 'of', 'SQL', 'are', 'dependent', 'on', 'text', 'lines', 'We', 'can', 'use', 'a', 'single', 'SQL', 'statement', 'on', 'one', 'or', 'multiple', 'text', 'line', 'Using', 'the', 'SQL', 'statements', 'you', 'can', 'perform', 'most', 'of', 'the', 'actions', 'in', 'a', 'database', 'SQL', 'depends', 'on', 'tuple', 'relational', 'calculus', 'and', 'relational', 'algebra', 'SQL', 'process', 'When', 'an', 'SQL', 'command', 'is', 'executing', 'for', 'any', 'RDBMS', 'then', 'the', 'system', 'figure', 'out', 'the', 'best', 'way', 'to', 'carry', 'out', 'the', 'request', 'and', 'the', 'SQL', 'engine', 'determines', 'that', 'how', 'to', 'interpret', 'the', 'task', 'In', 'the', 'process', 'various', 'components', 'are', 'included', 'These', 'components', 'can', 'be', 'optimization', 'Engine', 'Query', 'dispatcher', 'etc', 'non', 'SQL', 'queries', 'are', 'handled', 'by', 'the', 'classic', 'query', 'engine', 'but', 'SQL', 'query', 'engine', 'won', 't', 'handle', 'logical', 'files', 'DBMS', 'SQL']

In [7]: word_frequencies = {}
for word in doc:
    if word.text.lower() not in stopwords:
        if word.text.lower() not in punctuation:
            if word.text not in word_frequencies.keys():
                word_frequencies[word.text] = 1
            else:
                word_frequencies[word.text] += 1

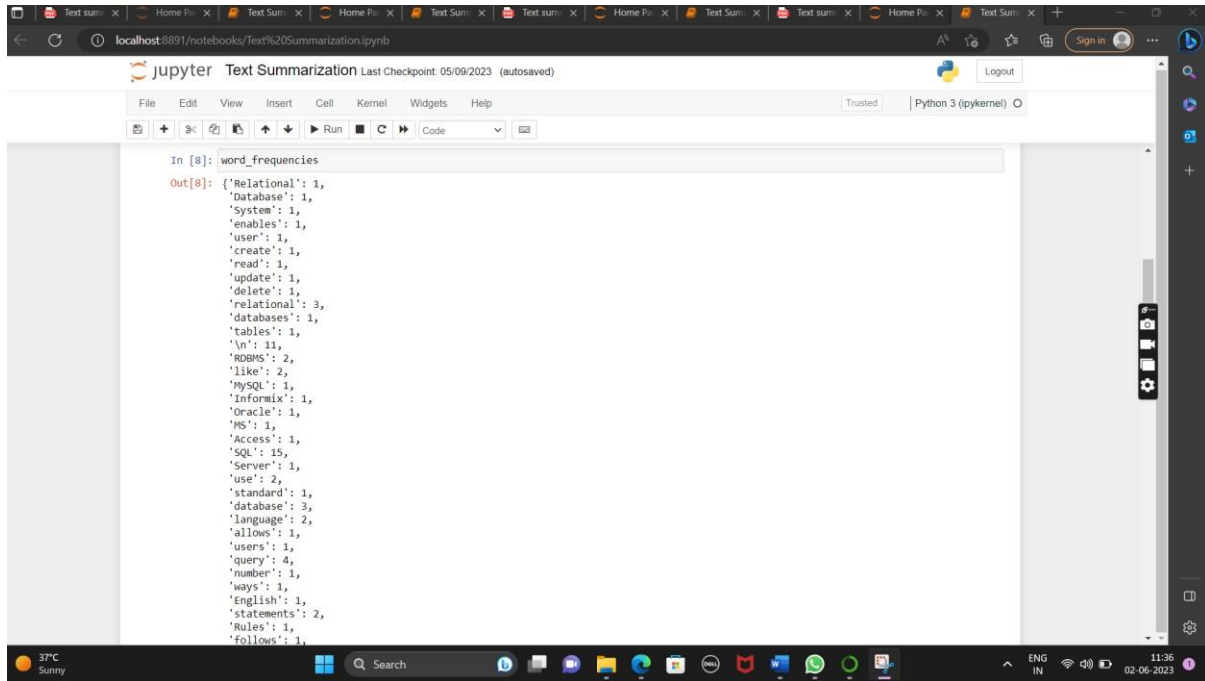
In [8]: word_frequencies

Out[8]: {'Relational': 1,

```

Screenshot 2: INPUT screen

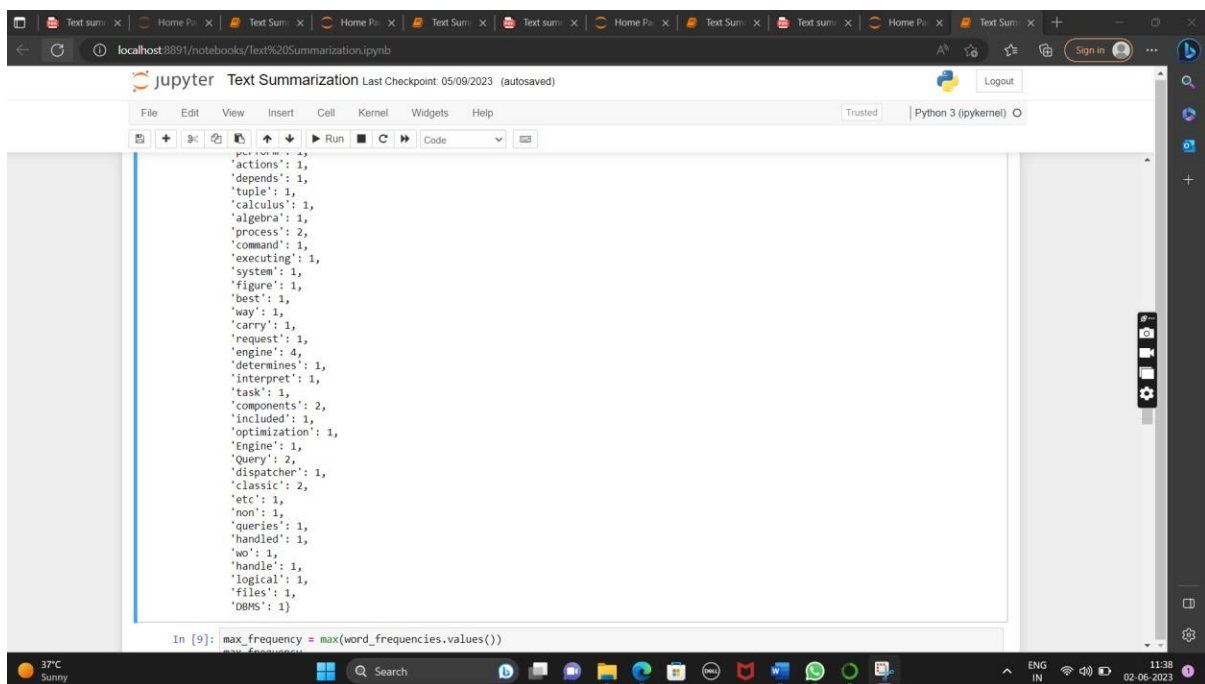
The input module and output spaces ins proceeded in this screenshot ,here we used token.text for dividing the text into tokens.



```

In [8]: word_frequencies
Out[8]: {'Relational': 1,
         'Database': 1,
         'System': 1,
         'enables': 1,
         'user': 1,
         'create': 1,
         'read': 1,
         'update': 1,
         'delete': 1,
         'relational': 3,
         'databases': 1,
         'tables': 1,
         '\n': 11,
         'RDBMS': 2,
         'like': 2,
         'MySQL': 1,
         'Informix': 1,
         'Oracle': 1,
         'MS': 1,
         'Access': 1,
         'SQL': 15,
         'server': 1,
         'use': 2,
         'standard': 1,
         'database': 3,
         'language': 2,
         'allows': 1,
         'users': 1,
         'query': 4,
         'number': 1,
         'ways': 1,
         'English': 1,
         'statements': 2,
         'Rules': 1,
         'follows': 1,
    }

```



```

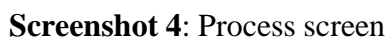
'actions': 1,
'depends': 1,
'tuple': 1,
'calculus': 1,
'algebra': 1,
'process': 2,
'command': 1,
'executing': 1,
'system': 1,
'figure': 1,
'best': 1,
'way': 1,
'carry': 1,
'request': 1,
'engine': 4,
'determines': 1,
'interpret': 1,
'task': 1,
'components': 2,
'included': 1,
'optimization': 1,
'Engine': 1,
'query': 2,
'dispatcher': 1,
'classic': 2,
'etc': 1,
'non': 1,
'queries': 1,
'handled': 1,
'wo': 1,
'handle': 1,
'logical': 1,
'files': 1,
'DBMS': 1}

In [9]: max_frequency = max(word_frequencies.values())

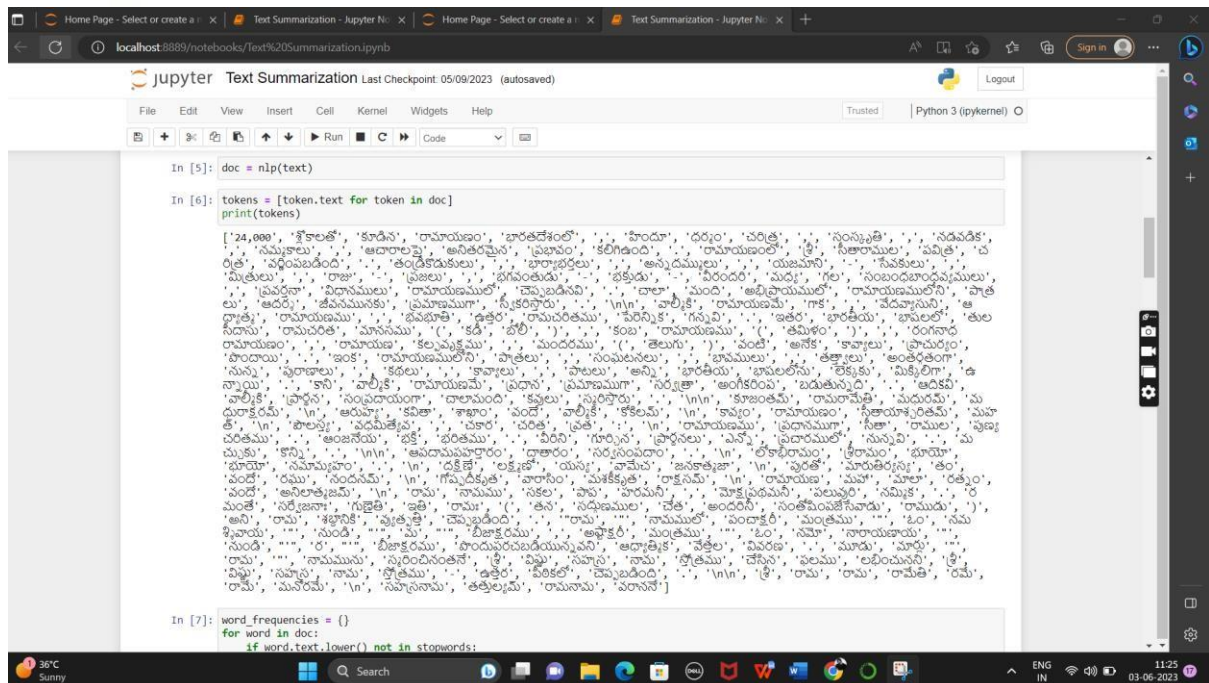
```

Screenshot 3 : OUTPUT screen 1(grammar check &spell check)

The pre-viewed text is displayed at the below the module with correct grammar and spell check



Page 537



OUT PUTS OF TEXT SUMMARIZATION IN TELUGU

5.CONCLUSION

We have seen that text summarization plays a critical role in saving users' time and resources as a result of the abundance of data available. Text summarization is a useful tool in today's world. Different algorithms and techniques have been used for this purpose in the past. These techniques, both separately and collectively, produce various types of summaries. To determine which summaries are better and shorter, compare their accuracy scores. The ROGUE score has been applied more frequently for this purpose. Similarly, TF_IDF scores have also been applied in some instances. The

summaries produced using these techniques are not always accurate. It can also be unrelated to the original document at times. As a result, research on this subject is ongoing, and various works have been done. There is no one model that consistently produces the best summaries. In order to produce more accurate summaries in the future, the models discussed can be modified. We could, for instance, use transfer learning and GANs. In the future, this may provide a means of creating and improving additional concepts for text summarization.



REFERENCES

- [1] Thailand Motor Vehicle Registered. CEIC Data Global Database.
- [2] Linda, S., Can public transport compete with the private car? Iatss Research, 2003. 27(2): p. 27-35.
- [3] CO2 emissions (metric tons per capita) - Thailand. THE WORLD BANK.
- [4] Cohen, A.J., et al., Estimates and 25-year trends of the global burden of disease attributable to ambient air pollution: an analysis of data from the Global Burden of Diseases Study 2015. The Lancet, 2017. 389(10082): p. 1907-1918.
- [5] Litman, T. and D. Burwell, Issues in sustainable transportation. International Journal of Global Environmental Issues, 2006. 6(4): p. 331-347.
- [6] Liu, M. and N. Choosri.. A technical solution to improve the red cab for touring in Chiang Mai: Chinese tourists' perspective. in 2016 Chinese Control and Decision Conference (CCDC). 2016. IEEE
- [7] Farooq, M.U., A. Shakoor, and A.B. Siddique. GPS based Public Transport Arrival Time Prediction. in 2017 International Conference on Frontiers of

Information Technology (FIT). 2017. IEEE.

[8] Bin, Y., Y. Zhongzhen, and Y. Baozhen, Bus arrival time prediction using support vector machines. Journal of Intelligent Transportation Systems, 2006. 10(4): p. 151-158.

[9] Maiti, S., et al. Historical data based real time prediction of vehicle arrival time. in 17th International IEEE Conference on Intelligent Transportation Systems (ITSC). 2014. IEEE.

[10] Fan, W. and Z. Gurmu, Dynamic travel time prediction models for buses using only GPS data. International Journal of Transportation Science and Technology, 2015. 4(4): p. 353-366.

Author's Profiles



Ms.M.ANITHA completed her Master of Computer Applications and Master of Technology. Currently working as an Assistant professor in the Department of



IJARST

International Journal For Advanced Research In Science & Technology

A peer reviewed international journal

ISSN: 2457-0362

www.ijarst.in

Masters of Computer Applications in the SRK Institute of Technology, Enikepadu, Vijayawada, NTR District. Her area of interest includes Machine Learning with Python and DBMS.



Ms.K.PAVANI completed her Master of Computer Applications. Currently working as an Assistant professor in the Department of Master of Computer Applications at SRK Institute of Technology, Enikepadu, NTR (DT). His areas of interest include Artificial Intelligence and Machine Learning.



Ms.G.MAHATHI SAI PRIYA is an MCA student

in the Department of Master Of Computer Applications at SRK Institute of Technology, Enikepadu, Vijayawada, NTR District. She has Completed Degree in B.Sc (Chemistry) from S.V.L Kranthi Degree College(Affiliated to Krishna University),Avanigadda. Her areas of interest are DBMS, JavaScript, Blockchain, Machine Learning with Python, HTML, CSS, Bootstrap and Django.