ISSN 1870-4069

Literature Summary Based on TF-IDF with Term-Term Correlation Matrix Analysis

Cybele Neves-Moutinho, María Fernanda Arámburo-Castell

Benemérita Universidad Autónoma de Puebla, Facultad de Cs. de la Computación, Puebla, Pue., México
{cybele.neves,maria.arambulo}@alumno.buap.mx

Abstract. This paper presents text summarization using the TF-IDF method, with preprocessing in the extraction of relevant information. The proposed technique is based on the identification of the most significant sentences within a document, evaluating the frequency of the terms and their rarity in the general context of the text. A preprocessing process is implemented that includes tokenization, where the text is divided into meaningful tokens, and the elimination of stopwords, which are terms that do not provide significant semantic value. Through a comparison between texts processed with and without the elimination of stopwords, it is shown that the inclusion of this step improves the generated summaries. The results indicate that the use of stopwords can introduce noise in the analysis, while their elimination allows the model to focus on the terms that truly reflect the essential content of the text, thus optimizing the quality of the summary.

Keywords: TF-IDF, Correlation Matrix, Abstractive Summarization, Extractive Summarization.

1 Introduction

Natural language processing (NLP) is a branch of artificial intelligence that makes human-computer interaction through a natural language in which the system can be able to understand, interpret, and generate human language so that it can interact effectively with people. This field covers applications such as machine translation, sentiment analysis, virtual assistants, and text processing [6]. Text processing is an area that focuses on analyzing, structuring, and interpreting large volumes of text data, such as documents, articles, social media posts, etc. This analysis allows for extracting patterns, identifying topics, and generating effective summaries of the information.

Information retrieval seeks to identify, classify, and extract important data from a large corpus. It filters unnecessary or redundant information, giving priority to elements that are more important to the user. Some information retrieval methods range from basic approaches, such as keyword search, to more advanced techniques, such as semantic and contextual relationships within texts [10].

Some of the most well-known methods are word frequency-based models such as Bag-of-Words, distributed representation models such as Word2Vec, probabilistic such as Vector Space Model (VSM) to calculate similarities [3], deep learning-based models such as Transformers, for example, BERT [8]. Some traditional information retrieval techniques include methods such as the inverted index, used in search engines, and also similarity measures such as cosine similarity to identify documents related to a query.

The Term Frequency - Inverse Document Frequency (TF-IDF) model is one of the most widely implemented methods for text processing and information retrieval [1], measuring the relative importance of a word within a document and in relation to an entire corpus.

Term Frequency (TF). Represents the number of times a term appears in a document, reflecting its local relevance.

Inverse Document Frequency (IDF). Penalizes common terms across all documents, highlighting those that are rarer.

The result is a weighted score that helps identify terms that are key in the context of the corpus. TF-IDF is widely used in summary generation, document classification, and building basic search engines.

2 Related Work

In comparative studies on the effectiveness of feature extraction techniques in text processing, authors Matata Das et al. [3] explored sentiment analysis using TF-IDF and N-Grams on IMDB movie review datasets and Amazon Alexa devices. Using classifiers such as Random Forest, SVM, and Logistic Regression, their results showed that TF-IDF consistently outperformed N-Grams, achieving a peak accuracy of 93.81% and an F1-score of 91.99% with the Random Forest classifier. On the other hand, Tawil et al. [8] compared the capabilities of TF-IDF, Word2Vec, and BERT in detecting phishing emails. While TF-IDF and Word2Vec achieved fairly good metrics such as F1-scores of 0.98 and accuracy rates of 97%, BERT proved to be superior, achieving an accuracy and F1-score of 0.99. This study shows the importance of advanced pre-trained models in complex tasks, such as cybersecurity, highlighting how traditional approaches such as TF-IDF remain competitive in certain applications, but are left behind by newer models in more complex scenarios.

Hans Christian et al. [2] present a study on automatic text summarization, focusing on a single document summarization approach using TF-IDF algorithm. The researchers developed a program capable of summarizing multiple documents, although the primary focus was on single document performance. The methodology involved preprocessing steps such as stop word removal, word tagging, and stemming to enhance the accuracy of the summarization process. The system utilized various features, including term frequency, inverse document frequency, and sentence characteristics, to identify relevant sentences for the summary. The results demonstrated that the TF-IDF algorithm achieved a summary accuracy of 67%, outperforming other online summarization tools. It highlights possible improvements to its results, such as skewing abstracts based on document titles and expanding the dataset.

A.Yugandhar et al. [9] realize automated text summarization using the TextRank algorithm, a graph-based ranking method employed in NLP. They developed a system that condenses lengthy texts into concise summaries while preserving the essential information and coherence of the original content. Techniques utilized include text preprocessing, sentence tokenization, and content overlap-based sentence scoring, with tools such as Python, NLTK, SpaCy, and NetworkX. The evaluation of the summarization system involved metrics like ROUGE scores, demonstrating that the generated summaries were comparable to those created by humans in terms of content and readability. The project underscores the effectiveness of the TextRank algorithm and suggests further enhancements through advanced NLP techniques and larger datasets for future research.

Several studies have explored improvements to the traditional TF-IDF model. Liang-Ching Chen et al. [1] proposed an extended TF-IDF method that assesses keyword relevance using intra-corpus comparisons with filtering mechanisms, applied to 20 scientific articles on climate change, showing better performance than traditional approaches. Cai-zhi Liu et al. [5] integrated Word2vec to generate vector representations of terms and refine TF-IDF calculations, adding factors such as keyword dispersion and class association, with a dataset of 3,000 texts across 11 topics, they reported gains in precision, recall, and F-measure. These studies show that modifications to TF-IDF can enhance its capacity to capture term relationships.

3 Proposed Method

Text summarization using the TF-IDF method allows to extract and retain important information by identifying the most relevant sentences. This approach is based on evaluating the importance of terms within a text, highlighting those that appear with high frequency in a sentence, but are rare in the rest of the document. This ensures that the summary reflects the most important content, while maintaining the general meaning of the original text.

The workflow in figure 1 illustrates the proposed method for the literary text summary processing. The input is a plain text document, followed by the preprocess, where the tokenization by sentences is performed, sequencing the extraction of characteristics by removing the stopwords and non-alphanumeric characters. The TF-IDF weight calculations begin, to later perform the calculations for each sentence, with this being able to generate the summary.

3.1 Preprocessing and Feature Extraction

The process starts with a plain text input. This plain text goes through a preprocess, tokenization is the initial step of text processing, in which its is divided into individual words or phrases, called tokens. These tokens can be categorized

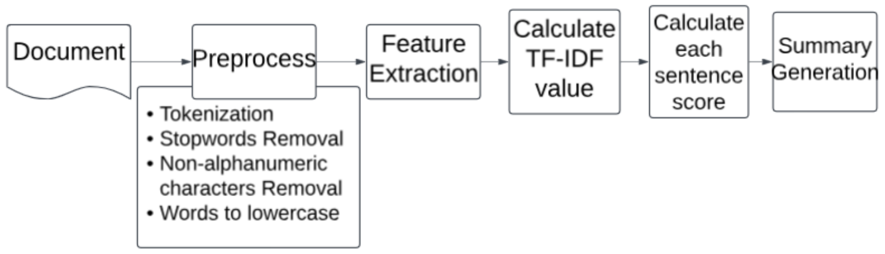


Fig. 1. Methodology of the literary text summary process.

into significant types for further processing, such as words, numbers, punctuation marks, and symbols. The level of detail in this step varies depending on the intended processing and available computational resources. In some cases, specific features of tokens, like capitalization or proximity to punctuation, may be retained, while in others, a simpler method may be used that focuses solely on contiguous alphabetical characters, as seen in many search engines [4].

Regular expressions (regex) are related to the concept of tokens in text processing. Since tokens can include words, numbers, punctuation marks, and other symbols, the goal of tokenization is to break up text into manageable pieces that can be further analyzed or processed, regular expressions provide a formal way to define the patterns that identify these tokens. They can also be used to categorize different types of tokens based on their patterns words, numbers, punctuation.

In this model, the tokenization functions are divided into two: sentence tokenization and word tokenization. First, sentence tokenization is performed to split the paragraph into sentences. For normalization with feature extraction, word tokenization is applied to split the written language string into words and punctuation.

Stopword removal is terms that frequently appear in a language but, on their own, do not contribute significant meaning to the content of the text, identified as articles, adverbs, conjunctions, pronouns and prepositions, auxiliary verbs. The removal of these words is done to reduce noise in text analysis. By removing words that do not contribute to the meaning of the content, the efficiency and accuracy of the model can be improved when processing the text, such as generating summaries.

The process usually involves the use of predefined lists of stopwords that can be customized based on the context of the analysis. By filtering out these words, the model is allowed to focus on more relevant terms that truly reflect the content and topic of the text. This model uses stopwords for texts in the Spanish language.

Word normalization by converting to lowercase is the process of transforming all letters in a text into their lowercase form. This process is done to prevent variations in case from affecting the analysis. In many cases, uppercase and lowercase words are considered equivalent in the context of text analysis.

Non-alphanumeric character removal involves removing from the text any character that is not a letter (a-z, A-Z) or a number (0-9). This includes punctuation marks, special symbols, extra whitespace, and other characters that do not contribute to the meaning of the text. This process is done to clean up the text and reduce noise that can interfere with the analysis. Non-alphanumeric characters often do not provide useful information for tasks such as text classification, sentiment analysis, or summary generation. By removing these characters, it makes it easier to extract relevant features and improves data quality.

The TF-IDF algorithm is used to measure the relevance of a word in a document relative to a set of documents. It consists of two parts: term frequency (TF) and inverse document frequency (IDF).

-Term Frequency (TF). Term frequency measures how many times a word appears in a document. It is calculated as:

$$TF(t,d) = \frac{n(t,d)}{\sum_k n(k,d)}. \tag{1}$$

Where:

– $n(t,d)$ is the number of times the term t appears in document d.
– $\sum_k n(k,d)$ is the sum of the frequencies of all terms in document d.

- Inverse Document Frequency (IDF). The inverse document frequency measures the importance of a term in the corpus. It is calculated as:

$$IDF(t) = \log\left(\frac{|D|}{|\{d \in D : t \in d\}|}\right). \tag{2}$$

Where:

– $|D|$ is the total number of documents in the corpus.
– $|\{d \in D : t \in d\}|$ is the number of documents containing the term t.

-TF-IDF calculation. Finally, the TF-IDF value of a term t in a document d is calculated by multiplying TF and IDF:

$$TF - IDF(t,d) = TF(t,d) \times IDF(t). \tag{3}$$

Term-term correlation is a technique used to measure the relationship between two terms in a set of documents. A common way to calculate this correlation is by cosine similarity, which measures the angle between two vectors in a vector space.

The cosine similarity between two vectors **A** and **B** is defined as:

$$Similarity(\mathbf{A},\mathbf{B}) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}. \tag{4}$$

Where:

– $\mathbf{A} \cdot \mathbf{B}$ is the dot product of the vectors $\mathbf{A}$ and $\mathbf{B}$.
– $\|\mathbf{A}\|$ is the norm (or magnitude) of the vector $\mathbf{A}$.
– $\|\mathbf{B}\|$ is the norm (or magnitude) of the vector $\mathbf{B}$.

The dot product of two vectors $\mathbf{A}$ and $\mathbf{B}$ is calculated as [7]:

$$\mathbf{A} \cdot \mathbf{B} = \sum_{i=1}^{n} A_i B_i. \tag{5}$$

where A_i and B_i are the components of the vectors $\mathbf{A}$ and $\mathbf{B}$, respectively. The norm calculation of a vector $\mathbf{A}$ is calculated as:

$$\|\mathbf{A}\| = \sqrt{\sum_{i=1}^{n} A_i^2}. \tag{6}$$

4 Experiments

The experiment aims to rank and select the most relevant sentences from a text based on their TF-IDF scores, for the generation of a summary. The process is detailed below based on the text and code provided:

– **Sentence score calculation.** Each sentence in the text is given a score that is calculated as the average of the TF-IDF values of the words that compose it. This score reflects the importance of the sentence, as words with high TF-IDF values tend to be less frequent in the document and more informative.
– **Sentence selection threshold.** The threshold is defined as the product of the mean of the sentence scores and a parameter called threshold_factor. If this value is equal to 1, the threshold corresponds exactly to the average of the sentence scores. Only sentences with a score equal to or higher than the threshold are included in the summary, ensuring that sentences with higher relative relevance are prioritized.
– **Balance in summarization.** Using a threshold_factor of 1 provides a balanced summary by including only sentences that have relevance at or above average. This ensures that the selected sentences provide meaningful information without being too restrictive or inclusive.
– **Adjusting the threshold.** The threshold_factor parameter can be modified to adjust the length and level of detail of the summary. For example:
  • A higher value results in shorter summaries, selecting only the most prominent sentences.
  • A lower value results in longer summaries, including sentences with scores close to the average.
– **Impact of document length on threshold.** The code includes an analysis of the impact of the threshold factor on summary length. By varying the *threshold_factor* over a range (e.g., from 0.5 to 1.5 in increments of 0.1), it is

observed how the summary length changes relative to the applied threshold. This allows the trade-off between precision and completeness in summarization to be explored.

The experiment design shows how the *threshold_factor* parameter and TF-IDF-based calculation allow for generating adaptive summaries. By varying the threshold, the length and relevance of the selected sentences are controlled, making the approach flexible for different contexts and needs.

The plain text used in this experiment was the Spanish literary archive Anecdotes of Nurses, by author Elisabeth G. Iborra. It is a book containing Mexican urban legends, nursing legends, colloquial vocabulary, and many short phrases like popular speech.

For the reported results, this project used a laptop with an Intel Core i7 10th-gen. processor, 16 GB RAM DDR4, and 512 GB of SSD. The software was implemented using Python and libraries for NLP with NLTK, Numpy, Math, SKLearn.

4.1 Running the Experiment

Running the experiment according to the code has several key steps, which are detailed below:

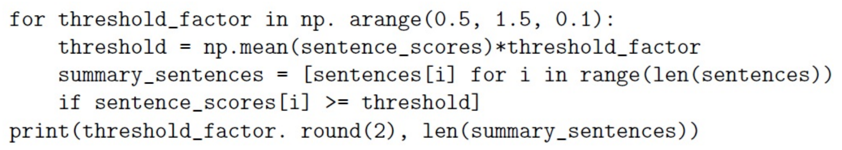
- **Obtaining the sentence scores**. The first step is to assign a score to each sentence in the text using TF-IDF. This score is calculated as the average of the TF-IDF values of all the words present in the sentence. To do this:
  - A previously generated TF-IDF matrix is used, where each word has a weight based on its frequency in the sentence and its rarity in the entire document.
  - The scores for each sentence are stored in a list `sentence_scores`.

  This allows prioritizing sentences with informative keywords, those that have high relevance within the document and low overall frequency.
- **Defining the threshold**. The threshold is determined by multiplying the average of the sentence scores by an adjustable parameter called the $threshold_factor$. This parameter controls the level of selectivity when including sentences in the summary: Here, `threshold_factor` is an adjustable parameter that defines how selective the model will be when including sentences in the summary:
  - If $threshold\_factor = 1$, sentences with scores equal to or higher than the average are included.
  - If $threshold\_factor > 1$, only sentences significantly higher than the average are included.
  - If $threshold\_factor < 1$, more sentences are included, even those close to the average.

– **Selecting sentences for the summary**. In this step, sentences are filtered according to the previously calculated threshold. Sentences with a score equal to or higher than the threshold are selected and stored in a list named `summary_sentences`. This ensures that only the most relevant sentences, based on their importance scores, are included in the final summary.
– **Generating the summary**. The selected sentences are then combined to create the final summary. This step forms a continuous text using only the most relevant sentences, based on their calculated importance scores.

4.2 Threshold Impact Analysis

The code includes additional analysis to see how adjusting the `threshold_factor` affects the summary length. Figure 2 of the code shows the iteration over a range of values for `threshold_factor` and calculating the summary length for each value.

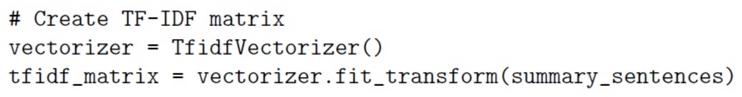


```
for threshold_factor in np. arange(0.5, 1.5, 0.1):
    threshold = np.mean(sentence_scores)*threshold_factor
    summary_sentences = [sentences[i] for i in range(len(sentences))
    if sentence_scores[i] >= threshold]
print(threshold_factor. round(2), len(summary_sentences))
```

Fig. 2. Threshold Factor Iteration and Summary Length Calculation.

This part of the code prints the length of the summary generated for each threshold value, allowing to analyze how the summary size changes with respect to different selectivity levels.

The code that implements the term-term similarity matrix. It aims to represent the selected sentences in a matrix where each vocabulary term has an associated weight. The weight is calculated using the TF-IDF formula previously demonstrated. The code made for this step is seen in figure 3.



```
# Create TF-IDF matrix
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(summary_sentences)
```

Fig. 3. The code step is where the TF-IDF matrix is created.

Here, `TfidfVectorizer` is a `scikit-learn` tool that generates the TF-IDF matrix from the sentences selected in `summary_sentences`.

Matrix transposition and term-term similarity calculation measure the similarity between terms in the text using cosine similarity across the columns of the transposed TF-IDF matrix. The matrix is transposed so that columns represent terms, and cosine similarity is then calculated between each pair of terms.

At the end of the entire execution, the summary of the plain text initially entered is presented as output.

5 Results

The literary text starts with 2637 sentences, which go through two different text preprocessing processes, one with the elimination of stopwords and another where the stopwords were kept, as observed in the table 1. In both cases, a different threshold factor of 0 to 1.4 is applied. The observed results are that the higher the threshold level, the more sentences are eliminated, which causes the meaning, coherence and concordance of the text to be lost, and the same for the other way around. The optimal balance is found at the threshold value of 1.

Table 1. Document length analysis based on threshold factor: comparison of text with and without stopword.

Threshold	Without stopwords	With stopwords
0	2637	2637
0.5	1564	1234
0.7	1087	780
1	**690**	**502**
1.2	551	391
1.4	438	317

Comparing the results of the threshold test, it is observed that the text with stopword obtained a better result and coherence for the summary.

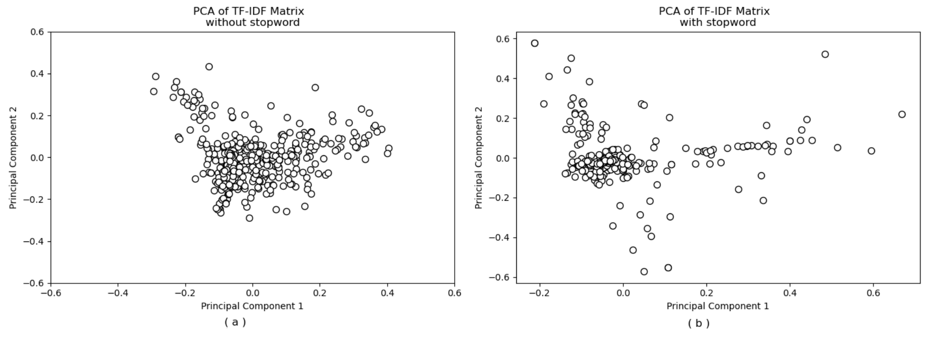


Fig. 4. PCA comparison with sentence points is their distance, which indicates their similarity.

PCA (Principal Component Analysis) reduces dimensions by projecting data along directions of highest variance, showing sentences as points whose distances reflect their similarity by TF-IDF. Figure 4 shows two PCA plots: (a) sentences without stopwords, (b) sentences with stopwords. Closer points indicate higher content similarity, which is greater when stopwords are kept.

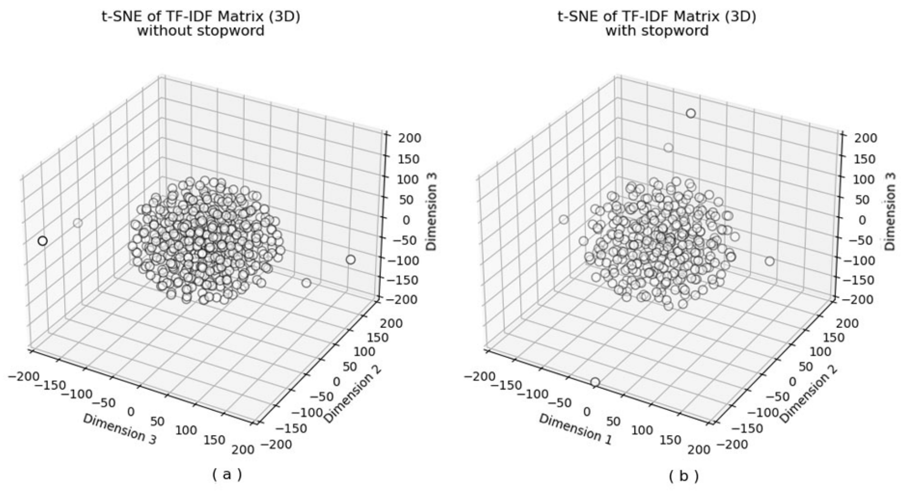


Fig. 5. t-SNE comparison with sentence points, which is their distance indicating the similarity between them.

Similarly in the figure 5 with the t-SNE (t-distributed Stochastic Neighbor Embedding) graph, which is a non-linear dimensionality reduction technique, suitable for incorporating high-dimensional data in a 3D space. We observe (a) sentences without stopwords and (b) sentences with stopwords.The plot shows that sentences kept stopwords exhibit greater clustering, indicating higher textual similarity.

An example of vocabulary from the text is seen in table 2 in the Term Term Matrix, with semantic relationships between terms having more value the rarer the word is in the sentence and the cosine similarity measures the similarity between two vectors as explained initially.

6 Conclusions

The proposed model demonstrates the effectiveness of the TF-IDF method in identifying relevant terms and generating text summaries, improving data quality through preprocessing techniques. However, when compared to advanced models such as BERT, it is evident that TF-IDF has limitations in capturing complex semantic relationships. The results indicate that, although TF-IDF is

Table 2. Term Term Matrix with semantic relationships between terms.

	doctor
apuntó	0.3759
desorientado	0.3759
doctor	**1**
jumo	0.1688
mire	0.0930
si	0.1506
simio	0.4039

useful, the integration of more sophisticated models could enhance the effectiveness in information extraction and language understanding, opening new opportunities for research in this area.

References

1. Chen, L.C.: An extended tf-idf method for improving keyword extraction in traditional corpus-based research: An example of a climate change corpus. Data Knowledge & Engineering **153**, 102322 (2024). https://doi.org/10.1016/j.datak.2024.102322
2. Christian, H., Agus, M., Suhartono, D.: Single document automatic text summarization using term frequency-inverse document frequency (tf-idf). ComTech: Computer, Mathematics and Engineering Applications **7**, 285 (12 2016). https://doi.org/10.21512/comtech.v7i4.3746
3. Das, M., K., S., Alphonse, P.J.A.: A comparative study on tf-idf feature weighting method and its analysis using unstructured dataset (2023), `https://arxiv.org/abs/2308.04037`
4. Ferilli, S.: Automatic digital document processing and management: Problems, algorithms and techniques. Springer Science & Business Media (2011)
5. Liu, C.z., Sheng, Y.x., Wei, Z.q., Yang, Y.Q.: Research of text classification based on improved tf-idf algorithm. Conference: 2018 International Conference on Network, Communication, Computer Engineering (NCCE 2018) pp. 218–222 (08 2018). https://doi.org/10.1109/IRCE.2018.8492945
6. Liu, W.: Chapter 1 - the essence of intelligence. In: Liu, W. (ed.) Integrated Human-Machine Intelligence, pp. 1–26. Elsevier (2023). https://doi.org/10.1016/B978-0-323-99562-7.00001-2
7. Manning, C.D.: Introduction to information retrieval. Cambridge University Press (2008)
8. Tawil, A.A., Almazaydeh, L., Qawasmeh, D., Qawasmeh, B., Alshinwan, M., Elleithy, K.: Comparative analysis of machine learning algorithms for email phishing detection using tf-idf, word2vec, and bert. Computers, Materials and Continua **81**(2), 3395–3412 (2024). https://doi.org/10.32604/cmc.2024.057279
9. Yugandhara Rao, A., Ratna Tezashri, K.and Kusum, K., Pydi Sai Rakesh, K., Manoj, M.: Text summarization using textrank algorithm. IJRTI: International Journal for Research Trends and Innovation **8**, 323 – 334 (7 2023)
10. Zhang, J., Li, J.: Chapter 14 - intelligent language knowledge for cognitive engine. In: Jianjun, Zhang, J.L. (ed.) Spatial Cognitive Engine Technology, pp. 187–197. Academic Press (2023). https://doi.org/10.1016/B978-0-323-95107-4.00012-3