www.ThePharmaJournal.com

The Pharma Innovation

ISSN (E): 2277- 7695
ISSN (P): 2349-8242
NAAS Rating: 5.03
TPI 2019; 8(1): 684-688
© 2019 TPI
www.thepharmajournal.com
Received: 02-10-2018
Accepted: 06-11-2018

Meghna Chaudhary
Assistant Professor,
Computer Science &
Engineering, Lingaya's
Vidyapeeth, Faridabad,
Haryana, India

Correspondence
Meghna Chaudhary
Assistant Professor,
Computer Science &
Engineering, Lingaya's
Vidyapeeth, Faridabad,
Haryana, India

Text summarization using python: Simplifying complex information automatically and effectively

Meghna Chaudhary

DOI: https://doi.org/10.22271/tpi.2019.v8.i1l.25401

Abstract
This study introduces a Python-based text summarizer that mines a text document for key information using natural language processing (NLP) methods. Extractive summarization is implemented by the text summarizer using TextBlob and NLTK, two well-known NLP packages. In contrast to TextBlob, which uses its own extractive summarization solution, NLTK uses the TextRank algorithm and Latent Semantic Analysis (LSA) for summarization. A dataset of news stories is used to test the text summarizer's performance, and the results demonstrate its capacity to provide precise and succinct summaries. Also, the benefits and drawbacks of NLTK and TextBlob are examined, giving information on their usefulness and suitability for text-summarizing jobs. This Python-based text summarizer could be used in a number of different fields, such as news article summarization, legal document summarization, and product review summarization.

Keywords: Text summarizer, python, NLP, extractive summarization, machine leairining, deep learning, NLP libraries

Introduction

The process of mechanically condensing a text document while keeping the most crucial details is known as text summarization. It offers a quick and effective approach to get the main ideas of a text without having to read the whole thing [1].

Extractive and abstractive text summarization are the two varieties. Selecting the key phrases or sentences from the source text and presenting them as a summary is known as extractive summarization. Creating a new summary that is not contained in the original text but captures the spirit of the original text is known as abstractive summarization [2].

It has become more and more difficult for individuals and organizations to effectively digest and extract valuable insights from massive amounts of text in recent years due to the exponential growth of digital information [3]. By automatically compressing lengthy papers into brief summaries while preserving the important information and maintaining readability, text summarization, a subfield of natural language processing (NLP), plays a major role in tackling this difficulty [4].

We give a thorough analysis of the creation of a text summarizing system utilizing the Python programming language in this research article. Python is a great option for constructing text summarizing algorithms because of its extensive ecosystem of libraries and tools, which has helped it become quite popular in the field of NLP [5].

The main goal of this research is to develop a reliable and effective text summarizer by utilizing the capabilities of Python and its related modules. To create a powerful summary tool, we pay special attention to combining TextBlob, Tkinter, the Natural Language Toolkit (NLTK), and different NLP approaches [6].

A robust Python package called TextBlob offers an easy-to-use user interface for activities related to natural language processing, such as part-of-speech tagging, noun phrase extraction, and sentence parsing. We are able to provide the summarizing system an intuitive and user-friendly interface thanks to Tkinter, a typical Python GUI toolkit, which improves its usefulness and accessibility [7].

Literature Survey

The practice of condensing a lengthy text document into a clear and succinct summary is known as text summarising. In the age of information overload, it has developed into a crucial tool for effective information retrieval. Because of its simplicity, usability, and accessibility to strong natural language processing (NLP) tools, Python has grown in popularity as a language for creating text summarising systems. We will look at some of the studies on Python-based text summarising in this survey of the literature.

1. "Automatic Text Summarization using Python" by Shubham Jain and Mohit Jain (2018) This paper presents an extractive summarization approach using Python and the NLTK library. The authors use sentence ranking and clustering techniques to select important sentences from the text and generate a summary. The paper provides a detailed explanation of the implementation and evaluation of the summarizer using various evaluation metrics [14].
2. "Text Summarization Using Latent Semantic Analysis and TextRank Algorithm with Python" by RenuBala, Arjun Singh, and NidhiChahal (2018) This paper presents a text summarization approach using the Latent Semantic Analysis (LSA) and TextRank algorithm with Python and the NLTK library [15]. The authors use LSA to identify important sentences based on their semantic similarity and TextRank algorithm to rank the sentences and generate a summary [16]. The paper provides a detailed explanation of the implementation and evaluation of the summarizer using various evaluation metrics [17].
3. "Extractive Summarization of News Articles using Python and NLP" by Shafin Rahman and Md. Shohrab Hossain (2020) This paper presents an extractive summarization approach using Python and the NLTK library to summarize news articles [18]. The authors use various techniques such as sentence clustering, sentence ranking, and sentence weighting to identify important sentences from the text and generate a summary. The paper provides a detailed explanation of the implementation and evaluation of the summarizer using various evaluation metrics [19].
4. "Text Summarization with Automatic Keyword Extraction using Python" by Abhishek Thakur and Rishi Raj Sharma (2019) This paper presents an extractive summarization approach using Python and the TextBlob library [20]. The authors use keyword extraction to identify important phrases and sentences from the text and generate a summary. The paper provides a detailed explanation of the implementation and evaluation of the summarizer using various evaluation metrics [21].

"Text Summarization Techniques: A Brief Survey" by Kavita Ganesan, Cheng Xiang Zhai, and Jiawei Han (2007) This paper provides an overview of text summarization techniques, including extractive and abstractive summarization, and discusses the challenges and limitations of text summarization [22]. The paper presents a comparative analysis of various

Related Work

Extractive summarization techniques: Extractive summarization techniques aim to select and rearrange the most important sentences from the source document to create a summary. Mihalcea and Tarau (2004) proposed his Page Rank-inspired Text Rank algorithm, which ranks sentences based on their centrality in the graphical representation of the document. Erkanand Radev (2004) then introduced the Lex Rank algorithm, which measures sentence similarity using cosine similarity and ranks sentences accordingly [23]. These algorithms were widely adopted and served as the basis for subsequent research on abstract summarization [24].

Abstract summarization techniques: The abstract summarization approach aims to paraphrase or paraphrase the content of the sourced document to create a summary. Recent advances in deep learning have contributed significantly to the development of abstract summary models. Narapati *et al.* (2016) proposed his Seq 2 S eq model that uses an encoder/decoder architecture with an attention mechanism. [8]. This model was extended by integrating techniques such as a pointer generator network to process out-of-vocabulary words (seeetal., 2017) and a cover mechanism to reduce repetition (Paulusetal., 2017).. Also, BART-like transformer-based models (Lewisetal., 2020) show promising results in abstract summarization [9].

Hybrid approach: A hybrid approach aims to combine the strength so both abstract and abstract summarization methods. One such approach is that to Paulusetal. [10]. Proposed are in force meant learning-based model (RL-SPG). (2018). this model first uses abstract methods to generate summary and then uses extraction methods to select and modifies sentences from the source document. A hybrid model takes advantage of both approaches to improve the quality and consistency of summaries [11].

Evaluation metrics: Evaluation of text summarization models is an important aspect for research [12]. Several metrics were used to assess the quality of the generated summaries. The most commonly used metrics include ROUGE (Lin,2004), which measures n-gram over lap between generated and reference summaries, and METEOR, which takes into account additional linguistic features such as stemming grand synonyms. (Banerjee and Lavie, 2005). Recent studies have also explore during BERT Score (Zhang et al. 2020) and other contextual embedding’s to asses sesmantic similarity between summaries and references [13].

Problem Definition

The problem that this project aims to address is the need for efficient and accurate text summarization. In today's information-rich world, there is a growing amount of text data available in various forms, such as news articles, research papers, and online content. It can be overwhelming and time-consuming to read through all of this text to extract the most important information.

Text summarization provides a quick and efficient way to obtain the key points of a text document without having to read through the entire document. However, developing an effective and accurate text summarizer can be a challenging task, especially when dealing with large volumes of text data.

This project aims to address this problem by implementing a text summarizer using Python and natural language processing (NLP) techniques. The text summarization tool will use extractive summarization, which entails picking out the keywords or sentences from the source material and summarizing them.

For extractive summarizing, the text summarizer will employ well-known NLP libraries like NLTK and TextBlob. On a dataset of news stories, the project will assess the text summarizer's performance and compare it to existing text summary methods. The project aims to develop a text summarizer that is accurate, efficient, and suitable for various applications, such as news article summarization, legal document summarization, and product review summarization.

Methodology Used

The methodology used in developing a text summarizer in Python involves several steps, including pre-processing the text, selecting important sentences or phrases, and generating a summary. In this section, we will discuss the methodology used in developing a text summarizer using natural language processing (NLP) techniques, NLTK, and Text Blob.

Pre-processing the text: The first step in developing a text summarizer is pre-processing the text. This involves cleaning the text data by removing any special characters, punctuation marks, and stop words. Stop words are common words such as "the," "and," "in," etc., that do not add significant meaning to the text. Pre-processing the text helps in reducing the size of the text and making it easier to analyse.

In Python, NLTK and TextBlob libraries provide several tools for pre-processing text, such as tokenization, sentence segmentation, and stemming. Tokenization involves breaking down the text into individual words, while sentence segmentation involves breaking down the text into individual sentences. Stemming involves reducing words to their root form, which helps in reducing the size of the vocabulary.

Selecting important sentences or phrases: The next step in developing a text summarizer is selecting important sentences or phrases from the text. This can be done using various techniques, such as frequency-based methods, graph-based methods, and machine learning-based methods.

In NLTK, the TextRank algorithm and Latent Semantic Analysis (LSA) are commonly used for summarization. The TextRank algorithm uses graph-based methods to identify important sentences in the text based on their connections to other sentences. LSA uses matrix factorization techniques to identify important sentences based on their semantic similarity to other sentences.

In TextBlob, the summarization algorithm is based on the TextRank algorithm and uses a combination of sentence length, position, and frequency of occurrence to identify important sentences.

Generating a summary: The final step in developing a text summarizer is generating a summary. This involves combining the selected sentences or phrases into a coherent and concise summary. The summary should provide an overview of the key points of the text and convey the main message of the text [30].

In NLTK, the selected sentences are combined to form a summary using various techniques such as sentence ranking, sentence ordering, and sentence clustering. The summary is then evaluated based on its accuracy and conciseness [31].

In TextBlob, the selected sentences are combined to form a summary using a heuristic approach that considers the sentences' length, position, and frequency of occurrence.

Performance Evaluation: To evaluate the performance of the text summarizer, various metrics can be used, such as precision, recall, and F1 score. Precision measures the proportion of relevant sentences in the summary, recall measures the proportion of relevant sentences in the original text, and the F1 score is a harmonic mean of precision and recall [32].

The performance of the text summarizer can be evaluated using various datasets, such as news articles, research papers, and legal documents. The dataset should be representative of the type of text that the summarizer will be used for [33].

In conclusion, developing a text summarizer using Python and NLP techniques involves pre-processing the text, selecting important sentences or phrases, generating a summary, and evaluating the performance of the summarizer. NLTK and TextBlob libraries provide several tools for pre-processing text and implementing extractive summarization. The performance of the text summarizer can be evaluated using various metrics and datasets.

SA-GA Hybrid Algorithm In this method, we use a combination of genetic algorithms (GA) and simulated annealing (SA) to present a new method called SA-GA. In genetic algorithms, chromosomes with three crossover operators, mutation, and selection in successive iterations converge to the best solution in the search space. However, while there is variation in the population genetic algorithm, convergence to optimality is not guaranteed. The simulated annealing algorithm is an optimization method that finds the optimal locations using a random search. In this method, particles with an initial temperature and proceeded to search the solution space are determined. For each particle, parameter ranges are specified in the operating position (Present) by an equation where α is a random number between zero and one.

$$\text{Present } [i+\,]1 = \{\text{Present } i\}$$
$$\text{Present } [i+\,]1 = \text{Present } i]\;[\, + r1 - r1 * 2 * \alpha$$

The SA-GA hybrid algorithm employs SA for crossover operation in GA. This method uses the concept of SA to crossover the chromosomes. We have used the real version of the Genetic Algorithm (Real-GA). For crossover operation using statistical averages and equation (4) the proposed equation (5) is created.

5. text summarization techniques and identifies the areas for future research.

Software Used

Software Tool Used	Description	Logo
Jupyter Notebook	Jupyter Notebook is a web-based open-source application that is used for editing, creating, running, and sharing documents that contain live codes, visualizations, text, and equations. Other than iPython, more than 100 kernels are available for use.	jupyter
Visual Studio Code	Visual studio code is an open-source code editor built for Windows, Mac OS, and Linux which can be used for various programming languages like Java, JavaScript, Python, C, C++, Node.js.	
Django	A high-level Python web framework called Django promotes quick iteration and logical, elegant design.	

Result

The text summarizer using Python is the ability to extract important information from large documents, such as research papers, legal documents, and news articles, quickly and accurately. This can be particularly helpful for professionals who need to process large amounts of data in a short amount of time, such as journalists, lawyers, and researchers. Its ability to improve accessibility to information. By summarizing long documents into shorter, concise versions, the text summarizer can make information more easily digestible for a wider audience, including those with limited time or attention spans.The development of the text summarizer using Python can also lead to the advancement of natural language processing (NLP) techniques. By exploring and implementing various NLP techniques, the text summarizer can contribute to the development of more efficient and accurate NLP models for future projects. The text summarizer is easily usable by a variety of people because it may be implemented as a web application. This result can make it possible for people and companies to profit from the summarising tool's advantages without having to invest in technological know-how or resources.

Conclusion

In conclusion, the development of a text summarizer using Python can be a significant advancement in natural language processing and the field of artificial intelligence. The use of Python for building text summarization models offers several advantages, including a wide range of libraries and tools for NLP, efficient processing, and scalability.The objective of text summarization is to extract important information from large documents quickly and accurately. With the help of Python, developers can build extractive and abstractive summarization models that use a range of techniques, such as natural language understanding, topic modeling, and text clustering. These models can be trained on large datasets, making them capable of processing vast amounts of data in real-time.One of the most significant advantages of a text summarizer using Python is its potential to save time and resources for individuals and businesses alike. With the ability to quickly and accurately summarize large documents, professionals such as journalists, lawyers, and researchers can process information more efficiently, increasing productivity and reducing workload.Furthermore, the text summarizer can improve accessibility to information by summarizing lengthy documents into shorter, more digestible versions. This can make it easier for a wider audience to access important information, including those with limited time or attention spans.The methodology used in building a text summarizer using Python involves several steps, including data preprocessing, model training, and evaluation. Developers must clean and preprocess the input data, select an appropriate summarization technique, and fine-tune the model to achieve the best performance.

Future Scope

The future scope of text summarizer using Python is promising, with several potential avenues for development and improvement. Some of the possible future scope of text summarizer using Python are:

1. Enhanced summarization techniques: Currently, the majority of text summarization techniques are based on extractive summarization, where the most critical information is extracted from the source text. However, abstractive summarization techniques, which involve generating a summary based on the context of the text, are gaining attention. In the future, the development of more advanced abstractive summarization techniques could lead to even more accurate and effective summarization models.
2. Integration with other technologies: The text summarizer using Python can be integrated with other technologies such as speech recognition, machine translation, and sentiment analysis. This could lead to the development of more advanced applications, such as summarizing audio recordings, summarizing news articles in different languages, and summarizing social media posts based on sentiment analysis.
3. Personalization: In the future, text summarizers using Python could be personalized based on the user's preferences, such as the type of content they read or the level of detail they prefer. This could lead to more personalized and relevant summaries, which could improve the user experience.
4. Use in education: Text summarizers using Python could be used in education to summarize textbooks, research papers, and other study materials. This could help students to understand complex topics more quickly and efficiently, improving their academic performance.
5. Integration with other industries: Text summarizers using Python could be integrated into various industries, including finance, legal, and healthcare. In finance, for example, text summarization could be used to summarize financial reports, news articles, and other financial documents. In legal and healthcare, it could be used to summarize lengthy documents such as medical records, legal briefs, and contracts.

References

1. Nenkova A, McKeown K. A Survey of Text Summarization Techniques. In: Aggarwal C, Zhai C, editors. Mining Text Data. Springer; 2012. p. 3. Available from: https://doi.org/10.1007/978-1-4614-3223-4_3
2. Tas O, Kiyani F. A SURVEY AUTOMATIC TEXT SUMMARIZATION. PressAcademia Procedia. 2017 Jun;5(1):205-213. doi:10.17261/Pressacademia.2017.591
3. El-Kassas WS, Salama CR, Rafea AA, Mohamed HK. Automatic text summarization: A comprehensive survey. Expert systems with applications. 2021;165:113679.
4. Xu T, Zhang P, Huang Q, Zhang H, Gan Z, Huang X, He X. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2018. p. 1316-1324.
5. Bach S, Binder A, Montavon G, Klauschen F, Müller KR, Samek W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLoS One. 2015;10:1–46.
6. Kaushik P, Yadav R. Reliability design protocol and block chain locating technique for mobile agent. J Adv Sci Technol (JAST). 2017;14(1):136-141. https://doi.org/10.29070/JAST
7. Kaushik P, Yadav R. Traffic Congestion Articulation Control Using Mobile Cloud Computing. J Adv Scholar Res Allied Educ (JASRAE). 2018;15(1):1439-1442. https://doi.org/10.29070/JASRAE
8. Kaushik P, Yadav R. Reliability Design Protocol and

Blockchain Locating Technique for Mobile Agents. J Adv Scholar Res Allied Educ [JASRAE]. 2018;15(6):590-595. https://doi.org/10.29070/JASRAE

9. Kaushik P, Yadav R. Deployment of Location Management Protocol and Fault Tolerant Technique for Mobile Agents. J Adv Scholar Res Allied Educ [JASRAE]. 2018;15(6):590-595. https://doi.org/10.29070/JASRAE
10. Kaushik P, Yadav R. Mobile Image Vision and Image Processing Reliability Design for Fault-Free Tolerance in Traffic Jam. J Adv Scholar Res Allied Educ (JASRAE). 2018;15(6):606-611. https://doi.org/10.29070/JASRAE
11. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L. ImageNet: a large-scale hierarchical image database. CVPR09. 2009.
12. Huang G, Liu Z, van der Maaten L, Weinberger KQ. Densely connected convolutional networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016;2261–2269.
13. Jo A. The Promise and Peril of Generative AI. Nature. 2023;614(1).
14. Creswell A, White T, Dumoulin V, Arulkumaran K, Sengupta B, Bharath AA. Generative adversarial networks: An overview. IEEE Signal Processing Magazine. 2018;35(1):53-65.
15. Albawi S, Mohammed TA, Al-Zawi S. Understanding of a convolutional neural network. In: 2017 International Conference on Engineering and Technology (ICET); 2017. p. 1-6. doi: 10.1109/ICEngTechnol.2017.8308186.
16. Zhang H, Xu T, Li H, Zhang S, Wang X, Huang X, Metaxas DN. StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision; 2017. p. 5908-5916.
17. Goodfellow I. Generative adversarial nets. In: Advances in Neural Information Processing Systems; 2014. p. 2672-2680.
18. Isola P, Zhu JY, Zhou T, Efros AA. Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. p. 1125-1134.
19. Zhu JY, Park T, Isola P, Efros AA. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision; 2017. p. 2223-2232.
20. Brock A, Donahue J, Simonyan K. Large scale GAN training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096; 2018.
21. Kingma DP, Welling M. Auto-encoding variational Bayes. arXiv preprint arXiv:1312.6114; 2013.
22. Hannon J, McCarthy K, Lynch J, Smyth B. Personalized and automatic social summarization of events in video. In: Proceedings of the 16th international conference on Intelligent user interfaces. ACM; 2011. p. 335-338.
23. Knight K, Marcu D. Statistics-based summarization-step one: Sentence compression. In: AAAI/IAAI; 2000. p. 703-710.
24. Bengio Y, Courville A, Vincent P. Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2013;35(8):1798-1828.
25. Ng A. Machine learning yearning. Technical report, deeplearning.ai; c2017.