iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 12 **Issue:** III **Month of publication:** March 2024

DOI: https://doi.org/10.22214/ijraset.2024.58785

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue III Mar 2024- Available at www.ijraset.com

Summarization of Video using Audio

Pratiksha Yadav[1], Nimish Rajgure[2], Rohan Thoke[3], Sahil Mandore[4], Prof. Pallavi Bhaskare[5]
Department of Computer Engineering Smt. Kashibai Navale College of Engineering Pune, India

Abstract: ***Online education has emerged as a highly effective means of delivering quality education to students. Its popularity has increased due to the high-quality visual and graphical content, delivered by subject matter experts, and the convenience of learning anytime and anywhere. However, students may face time constraints that prevent them from fully engaging with the course content. To address this issue, video transcript summarizers have gained popularity. These tools extract the most important topics from a video, allowing students to understand the essence of the class without having to watch the entire video. Our system focuses on developing a module using txtai.pipeline with Python to summarize online class videos. We use Whisper, a general-purpose speech recognition model, to train our model on a large dataset of diverse audio. The model takes the URL of a video as input and uses two algorithms to summarize the content: TF-IDF and Gensim. The summarization process is subjective, and we have incorporated two prominent methods: cosine similarity and ROUGE score. The former does not require a human-generated summary for reference, while the latter does. Our results show that the efficiency obtained using cosine similarity is greater than 90% in both TF-IDF and Gensim cases. The efficiency obtained in the case of ROUGE score is between 40-50%.***

Keywords: Text summarization, model, video.

I. INTRODUCTION

In recent years, online education has become increasingly popular due to its ability to deliver high-quality education to students through expert-led courses with high-quality visual and graphical content, and the convenience of learning anytime and anywhere. However, students may face time constraints that prevent them from fully engaging with the course content. To address this issue, video transcript summarizers have gained popularity. These tools extract the most important topics from a video, allowing students to understand the essence of the class without having to watch the entire video. In this context, the use of advanced technologies such as Whisper and txtai.pipeline with Python has revolutionized the process of video to text summarization. Whisper, a general-purpose speech recognition model, is trained on a large dataset of diverse audio and is a multitasking model that can perform multilingual speech recognition, speech translation, and language identification. txtai.pipeline is a powerful tool that can be used to develop modules for summarizing online class videos. In this article, we will explore the use of Whisper and txtai.pipeline to summarize online class videos, and discuss the two algorithms used for summarization: TF-IDF and Gensim. We will also delve into the subjective nature of the summarization process and the two prominent methods used to evaluate the efficiency of the summarization: cosine similarity and ROUGE score.

II. VIDEO TO TEXT SUMMARIZATION MODELS

Video to text summarization models need to process both visual and audio information, requiring techniques such as video segmentation, object recognition, speech-to-text conversion, and natural language processing. Traditional text summarization models, on the other hand, only deal with textual data. Video to text summarization models are more complex due to the multimodal nature of the input data, and they often involve advanced machine learning and deep learning techniques to process and summarize the information from videos.

A. Subprocess

The subprocess module in Python is a powerful utility that allows you to spawn new processes, interact with their input/output/error streams, and obtain information about their execution. It provides a high-level interface for executing external commands or programs from within a Python script. The module enables you to interact with operating systems' command line interfaces (CLIs) and manage child processes created by executing shell commands. You can spawn new processes via the execution of system commands, connect to their input/output/error pipes, and obtain their return codes after completion. Using subprocess, you can perform a wide range of tasks, including running system commands, capturing their output, and managing child processes.

©IJRASET: All Rights are Reserved

The key functionalities and features of the subprocess module include spawning processes, input/output handling, capturing output, managing child processes, cross-platform compatibility, and more. The subprocess module is designed to work seamlessly across different operating systems (e.g., Windows, Linux, macOS), making it a versatile choice for executing system commands and managing processes in a platform-independent manner.
The subprocess module in Python is a powerful utility that allows you to spawn new processes, interact with their input/output/error streams, and obtain information about their execution.

B. Txtai.pipeline

The txtai.pipeline library is a Python library that simplifies the process of building natural language processing (NLP) pipelines. It provides an easy-to-use interface for performing common NLP tasks such as text summarization, question answering, and semantic search. The library uses state-of-the-art machine learning models to accomplish these tasks and can be easily integrated into existing Python projects. The library offers text summarization functionality, which can be used to generate concise summaries of longer documents or articles using various techniques such as Term Frequency-Inverse Document Frequency (TF-IDF), cluster-based summarization, neural network summarization, fuzzy logic summarization, and query-based summarization. The library also enables question answering functionality, allowing users to pose questions about a given text document and retrieve relevant answers. Additionally, the library supports semantic search, which can be used to discover relevant information within a corpus of text based on the meaning and context of the query. The library leverages state-of-the-art machine learning models for text processing tasks, ensuring high-quality results and accurate outputs. The txtai.pipeline library is open-source and can be easily installed using the pip package manager, making it accessible to Python developers. Overall, the txtai.pipeline library provides a convenient and powerful tool for implementing various NLP tasks using state-of-the-art machine learning models. Its user-friendly interface and comprehensive functionality make it a valuable addition to any Python project requiring NLP capabilities.
The pipeline helps to identify the best model for a particular task, removing the need for manual selection. TxtAi pipeline streamlines data ingestion, data outputs, and workflows for image and speech processing. ESPnet Library is the foundation for hosting text-to-speech and speech to text models. Text-to-speech pipeline is called text-to-speech, while speech to text is called transcription. Initiating the pipeline involves calling the class and providing the text for conversion. Displaying speech object using ipython.display method. Understanding the process of converting speech object to numbers for AI models. TxtAi Vision Speech processing can create workflows with multiple wave files and generate text for all files, making the process intuitive. Data processing pipelines work with CSV and text files, segmenting and extracting data using Tika or BeautifulSoup. Tabular pipeline takes a CSV file, like a COVID-related CSV file. Initiating tabular pipeline removes unnecessary columns and provides specific output like URL and title. TxtAi Vision Speech Data processing pipelines can extract various data from websites using text extractor. TxtAi Vision Speech Data processing pipelines can work with multiple URLs to create a workflow for data extraction. Caption pipeline returns a description of the image. Image hash creates a hash based on the data from the images. Object detection pipeline detects objects inside the image. Different models give different outputs for object detection. Image hashing provides unique identifiers for images and can be used for creating indexes. Vector embedding is different from hash indexing and cannot be used for searching. Image embedding and searching through image are separate processes and will be explained in a future video.

C. Pytube

PyTube offers native classes like YouTube class and Playlist class. Playlist class serves as an interface between the program and actual YouTube playlist, requiring input like playlist URL. Creating an instance by passing the URL of the playlist to the playlist class. Exploring additional functionalities such as checking description, views, title, channel information, and retrieving video URLs. The YouTube class allows access to video titles, views, and publish dates. Pytube is a Python library that is used to download videos from platforms like YouTube. Although Pytube does not directly relate to YouTube summarization, there are projects that utilize Pytube alongside other tools to automate the process of generating summaries from YouTube videos. These projects involve converting audio from videos into text through speech recognition, followed by applying text summarization algorithms. The specific example given in the search results utilizes Pytube to download the YouTube video, then employs Whisper for speech recognition, and finally applies BART for text summarization. The resulting summary is stored in a text file and displayed for review. These projects offer practical implementations for processing long-format content like YouTube videos and provide a means to access and analyze vast amounts of information without requiring extensive manual effort. However, the legality of scraping and repurposing content from YouTube should always be considered, and permission must be obtained where necessary.

Additionally, the quality of the output may vary depending on factors such as the accuracy of the speech recognition software and the effectiveness of the chosen text summarization algorithm. It is a lightweight, dependency-free, and user-friendly library that offers a simple way to retrieve YouTube videos for offline viewing or further processing. Pytube has several key aspects that make it a popular choice among developers. Firstly, it is dependency-free, which ensures ease of installation and compatibility across environments. Secondly, it supports both progressive and DASH streams, enabling users to select their preferred formats and qualities. Thirdly, it comes with a built-in command-line interface, which simplifies the download process even further. Fourthly, users can customize the output directory structure and filenames during the download process. Fifthly, Pytube outputs caption track files in .srt format, facilitating subtitling efforts. Sixthly, Pytube runs smoothly on Windows, macOS, Linux, and other UNIX-like operating systems. Lastly, Pytube boasts detailed documentation and an active community on GitHub, fostering collaboration and troubleshooting support. Although Pytube itself does not directly contribute to YouTube video summarization, it serves as a foundation for developing projects that leverage YouTube videos for downstream applications, such as summarizing video content. Such projects might integrate Pytube with speech recognition and text summarization technologies to create automated solutions for analyzing and synthesizing long-form video content. While pytube itself doesn't facilitate text summarization, it can serve as a foundation for creating projects that leverage YouTube videos for downstream applications. PyTube offers native classes like YouTube class and Playlist class. Playlist class serves as an interface between the program and actual YouTube playlist, requiring input like playlist URL. Creating an instance by passing the URL of the playlist to the playlist class. Exploring additional functionalities such as checking description, views, title, channel information, and retrieving video URLs. The YouTube class allows access to video titles, views, and publish dates .

D. Whisper

Whisper is a Python library developed by OpenAI that facilitates automatic speech recognition (ASR) tasks. It is designed to transcribe speech from audio files and provides an easy-to-use interface for developers. Whisper is built on an encoder-decoder transformer architecture, which is a type of neural network model commonly used in natural language processing tasks. This architecture enables Whisper to accurately transcribe speech from audio files into text. Whisper is an open-source project developed by OpenAI, and its source code is freely available for anyone to view, modify, and distribute. Whisper offers five different model sizes to accommodate various use cases and computational resources. These sizes include tiny, base, small, medium, and large, allowing users to choose the appropriate model based on their requirements. Whisper can be easily installed using the pip package manager, making it accessible to Python developers. Whisper requires Python 3.7 or later and a recent version of PyTorch, a popular deep learning library. Additionally, it relies on FFmpeg, an audio-processing library, which needs to be installed separately on your machine. Whisper is released under the MIT License, which permits users to freely use, modify, and distribute the library. Both the code and model weights of Whisper are available on GitHub, allowing for transparency and community contributions. Overall, Whisper is a versatile and user-friendly Python library for performing automatic speech recognition tasks. Its availability as an open-source project, along with its easy installation process and various model sizes, make it a valuable tool for developers working on speech-related applications.

E. Time

Pytube is a Python library that allows developers to interact with the YouTube API and download videos from YouTube using Python code. It's a useful library for working with YouTube content, but it's distinct from the time module in Python. The time module in Python provides various functionalities for working with time, including obtaining the current time, converting between different time representations, formatting time as strings, and calculating time differences. It's a fundamental module in Python's standard library and is widely used in various applications that involve time-related operations. The time module includes functions for obtaining the current time, such as time.time(), time.ctime(), and time.localtime(). It also includes functions for converting between timestamps and structured time representations, such as time.mktime() and time.strptime(). The structured time representation is provided via time.struct_time, which provides convenient access to individual time components. Additionally, the time module includes platform-independent functions, such as time.asctime() and time.strftime(), for formatting time as strings. Overall, the time module is a powerful tool for working with time in Python and is widely used in various applications.

III. METHODS FOR CONVERTING AUDIO INTO TEXT

Python offers several methods for converting audio into text, including Whisper, txtai.pipeline, and subprocess. Whisper is an automatic speech recognition model that is trained on 680,000 hours of multilingual data collected from the web.

It is robust to accents, background noise, and technical language, and it supports 99 different languages' transcription and translation from those languages into English. To use Whisper, one can install the library, load the model, and transcribe the audio file. The Hugging Face Transformers library also offers a pipeline feature that simplifies the implementation process and provides pre-trained models, including Open AI Whisper, that can be easily downloaded from the Hugging Face model hub. The pipeline feature facilitates easy customization of the model by allowing users to choose between tiny, base, or large models based on their specific requirements. To perform speech-to-text using Open AI Whisper and the Hugging Face Transformers pipeline feature, one can check for GPU availability, install the Transformers library, import the pipeline module, specify the task type and model, set the device for inference, provide the input audio file, and run the code to display the text output. Preprocessing audio files can help improve speech-to-text accuracy, and techniques such as noise reduction, audio normalization, and filtering can be applied to enhance the quality of input audio before passing it to the Open AI Whisper model. Other methods for converting audio into text in Python include using subprocess and the SpeechRecognition library, which can transcribe an audio file automatically. The program converts the audio file to WAV format, loads the audio file, and feeds it to a speech recognition system. Using txtai and Whisper for audio transcription and summarization offers several advantages. Whisper offers impressive accuracy, even for audio with multiple speakers and some background noise. Using txtai and Whisper can streamline workflows for content creators and marketers, reducing the time and effort required for transcription and content creation. txtai provides a range of summarization options, including extractive and abstractive summarization, allowing users to tailor the output to their needs. txtai and Whisper can be integrated with other tools and libraries, such as OpenAI's ChatGPT API, to enhance functionality and improve results. Whisper supports a variety of languages, making it a versatile tool for transcription and summarization in different contexts. All-in-one solution: txtai offers an all-in-one embeddings database for semantic search, LLM orchestration, and language model workflows, providing a range of features in a single package. Overall, using txtai and Whisper for audio transcription and summarization can improve accuracy, efficiency, and customization, making it a valuable tool for a range of applications. The Whisper API, based on the Whisper model, offers fast and accurate transcription services in over 50 languages and supports various audio formats, making it suitable for real-time audio transcription. The Whisper API can transcribe audio and return the entire transcript for the submitted audio as a single, contiguous text response, making it well-suited for real-time applications. Additionally, the Whisper API has a PAYG (pay-as-you-go) model, allowing users to transcribe audio without spending a fixed amount of money upfront. This makes it a convenient and cost-effective solution for real-time transcription needs. Therefore, Whisper can be used for real-time audio transcription. However, it's important to note that the real-time capability also depends on the infrastructure and resources available for running the API. As for summarization, while there is no direct evidence of real-time summarization using Whisper and txtai, txtai provides fast performance and can be integrated with other tools, which may enable real-time or near-real-time summarization when combined with Whisper's transcription capabilities.

IV. EXTRACTING SUMMARY FROM TEXT

Summarizing text effectively is a pivotal task in natural language processing (NLP) and information retrieval, driving the development of diverse techniques and methodologies aimed at distilling the core essence of the original content. In this method, key sentences or passages are cherry-picked from the original text based on criteria such as relevance, informativeness, and coherence. Techniques like graph-based algorithms (e.g., TextRank), machine learning models (e.g., Support Vector Machines, Random Forests), and neural networks (e.g., Recurrent Neural Networks, Transformer-based models) are employed to rank and select sentences, crafting a summary that mirrors the most salient information. Going beyond mere extraction, this approach generates novel sentences that encapsulate the primary ideas and concepts from the source text. Utilizing advanced natural language generation techniques, models such as sequence-to-sequence architectures with attention mechanisms or transformer-based frameworks like BERT (Bidirectional Encoder Representations from Transformers) adeptly paraphrase and rephrase input text, yielding coherent and concise summaries. Techniques like Latent Dirichlet Allocation (LDA) and Non-Negative Matrix Factorization (NMF) delve into the underlying themes or topics present in a corpus of text documents. By uncovering the principal topics, these methods facilitate the creation of summaries that hone in on the most pertinent and significant information. This method focuses on condensing sentences while preserving their core meaning. Through the elimination of redundant or extraneous information and simplification of complex structures, sentence compression ensures that only the essential content remains. It serves as both a precursor to extractive summarization and a standalone approach for crafting succinct summaries. Summaries to specific user queries or information needs is the hallmark of this approach. By pinpointing key terms or phrases in the query, relevant summary content is extracted or generated, ensuring alignment with the user's search intent. Query-based summarization proves particularly valuable in information retrieval systems and question-answering applications.

Ultimately, the selection of a summarization method hinges on factors such as the nature of the text data, desired abstraction level, and specific task objectives. By harnessing a blend of extractive and abstractive techniques alongside topic modeling and query-based approaches, summaries can effectively distill the essence of the original text, offering valuable insights to users. Utilize the Whisper API to transcribe the audio files into text. Whisper offers fast and accurate transcription services in over 50 languages and supports various audio formats, making it suitable for real-time audio transcription. Use txtai's Summary pipeline, which runs a text2text model that abstractively creates a summary of the input text. There are no specific techniques mentioned that txtai and Whisper use to extract text from images with complex backgrounds. However, there are some general techniques that can be used to extract text from images with complex backgrounds, preprocessing techniques like image enhancement, noise reduction, and contrast adjustment can help improve the quality of the image and make it easier to extract text from complex backgrounds. Segmentation techniques can help separate the text from the background by identifying the regions of the image that contain text. Optical character recognition (OCR) techniques can be used to recognize and extract text from the segmented regions of the image. Machine learning techniques like deep learning can be used to train models to recognize and extract text from images with complex backgrounds. It's important to note that the accuracy of text extraction from images with complex backgrounds depends on the quality of the image and the complexity of the background. While these techniques can help improve the accuracy of text extraction, they may not always be effective in all cases.

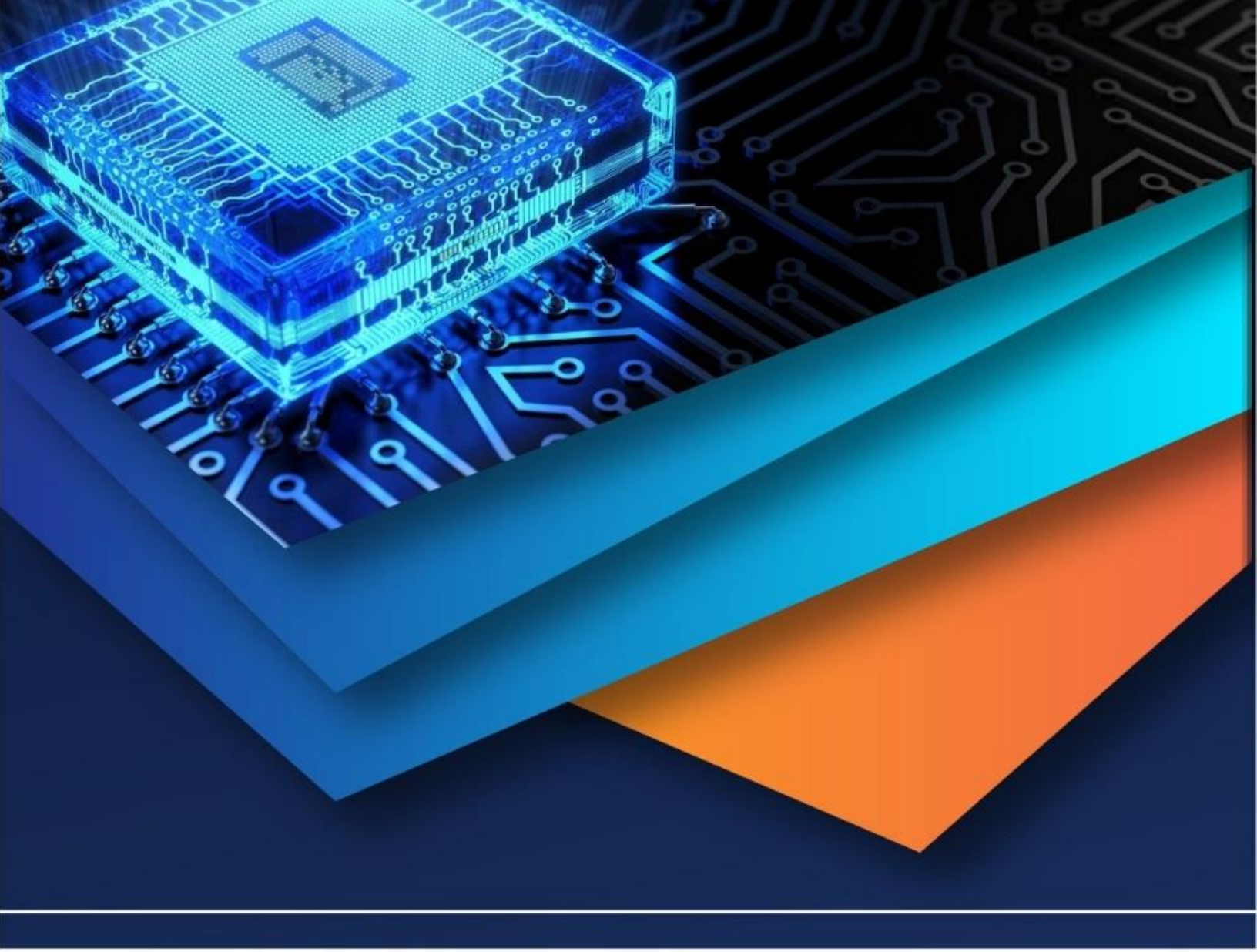
V. CONCLUSION

The project outlines a pioneering approach to multimodal summarization, which involves converting video data into audio and subsequently into text for efficient content analysis and summarization. The process begins with advanced speech recognition techniques to extract auditory information from video content. Next, state-of-the-art natural language processing algorithms convert the audio data into textual transcripts, capturing the semantic essence of the original video. This sequential conversion facilitates the integration of audio-visual data into a unified textual representation, enabling comprehensive content analysis and summarization. Leveraging the complementary strengths of audio and text modalities, the approach enhances the summarization process by incorporating diverse sources of information. Comparative analysis with existing multimodal summarization techniques demonstrates the effectiveness of the proposed method in generating concise and informative summaries across various types of video content. Overall, the study contributes to advancing the field of multimodal summarization by introducing a novel framework that exploits the synergies between audio and text modalities for comprehensive content analysis and summarization.

REFERENCES

[1] Wang, Y., Du, S., & Zhan, Y. (2008). Adaptive and Optimal Classification of Speech Emotion Recognition. 2008 Fourth International Conference on Natural Computation. doi:10.1109/icnc.2008.713
[2] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, in Proceedings of the IEEE, vol. 86, no. 11 (1998), pp. 2278–2324
[3] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu P. Kuksa, Natural language processing (almost) from scratch. J. Mach. Learn. Res. 12, 2493–2537 (2011)
[4] G. Erkan, D.R. Radev, Lexrank: graph-based lexical centrality as salience in text summarization. J. Artif. Intell. Res. 457–479 (2004)
[5] Y. Gong, X. Liu, Generic text summarization using relevance measure and latent semantic analysis, in Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (ACM, 2001), pp. 19–25
[6] C. Li, X. Qian, Y. Liu, Using supervised bigram-based ilp for extractive summarization, in Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL, 2013), pp. 1004–1013
[7] Y. Zhang, M.J. Er, R. Zhao, Multi-document extractive summarization using window-based sentence representation, in 2015 IEEE Symposium Series on Computational Intelligence (IEEE, 2015), pp. 404–410
[8] R. Nallapati, F. Zhai, B. Zhou, SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents, in The Thirty-First AAAI Conference on Artificial Intelligence (2016)
[9] M. Denil, A. Demiraj, N. Kalchbrenner, P. Blunsom, N. de Freitas, Modelling, visualising and summarising documents with a single convolutional neural network. In: arXiv preprint arXiv:1406.3830 (2014)
[10] A. Pardha Saradhi, A. Sai Kiran, A. Dileep Kumar, B. Srinivas, M.V. Nageswara Rao, Design and implementation of speech to text conversion on raspberry Pi, in International Journal of Innovative Technology and Exploring Engineering (IJITEE), vol. 8, Issue-6. ISSN: 2278-3075, April 2019
[11] S. Jadon, M. Jasim, Unsupervised video summarization framework using keyframe extraction and video skimmingar. Xiv preprint. arXiv:1910.04792v2 (2020)
[12] Z. Ji, K. Xiong, Y. Pang, X. Li, Video summarization with attention-based encoder-decoder networks. arXiv preprint arXiv:1708.09545v2 (2018)
[13] Y. Zhang et al., Extractive document summarization based on convolutional neural networks, in IECON 2016—42nd Annual Conference of the IEEE Industrial Electronics Society (2016), pp. 918–922
[14] Y. Kim, Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
[15] P. Sharif, BBC News Summary, kaggle.com (2021). https://www.kaggle.com/pariza/bbc-news-summary. Accessed 26 Feb 2021
[16] D. Greene, P. Cunningham, Practical solutions to the problem of diagonal dominance in kernel document clustering, in Proceedings of the ICML 2006 (2006)

[17] Beigi, M., Chang, S.-F., Ebadollahi, S., & Verma, D. (2009). Muti-scale temporal segmentation and outlier detection in sensor networks. 2009 IEEE International Conference on Multimedia and Expo. doi:10.1109/icme.2009.5202496
[18] Fukuzawa, K., Komori, Y., Sawai, H., & Sugiyama, M. (1992). A segment-based speaker adaptation neural network applied to continuous speech recognition. [Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing. doi:10.1109/icassp.1992.225879
[19] Tanaka, T. (n.d.). Spatio-temporal pattern recognition by competitive networks. Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan). doi:10.1109/ijcnn.1993.716809
[20] Wang, S., Chen, Z., & Ding, Z. (2019). The Unified Object Detection Framework with Arbitrary Angle. 2019 5th International Conference on Big Data and Information Analytics (BigDIA).
[21] Dulani Meedeniya, Isuru Ariyarathne, Meelan Bandara, Roshinie Jayasundara, Charith Perera, "A Survey on Deep Learning Based Forest Environment Sound Classification at the Edge", ACM Computing Surveys, vol.56, no.3, pp.1, 2024.
[22] Pencea Maria Larisa;Ruxandra Tapu 2022 International Symposium on Electronics and Telecommunications (ISETC)
[23] Tatevik Ter-Hovhannisyan;Karen Avetisyan 2022 Ivannikov Memorial Workshop (IVMEM)
[24] Zhaoliang Gu;Mengzhao Zhu;Wenbing Zhu;Ran Xu;Jiabin Zhou;Jian Wang;Qingdong Zhu
[25] M -S. Chaouche;H. Houassine;S. Moulahoum;S. Bensaid;D. Trichet2019 19th International Symposium on Electromagnetic Fields in Mechatronics, Electrical and Electronic Engineering (ISEF)
[26] Soo Ye Kim;Jeongyeon Lim;Taeyoung Na;Munchurl Kim 2019 IEEE International Conference on Image Processing (ICIP)
[27] Hongxiang Fan;Ho-Cheung Ng;Shuanglong Liu;Zhiqiang Que;Xinyu Niu;Wayne Luk 2018 28th International Conference on Field Programmable Logic and Applications (FPL)
[28] Anton Saveliev;Mikhail Uzdiaev;Malov Dmitrii 2019 12th International Conference on Developments in eSystems Engineering (DeSE)
[29] Kartik Hegde;Rohit Agrawal;Yulun Yao;Christopher W Fletcher 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)
[30] Martin Kaloev;Georgi Krastev 2021 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)
[31] Le Lyu;Yang Shen;Sicheng Zhang 2022 IEEE International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA)
[32] Weihua He;Yongyun Wu;Xiaohua Li 2021 IEEE 5th Information Technology,Networking,Electronic and Automation Control Conference (ITNEC)
[33] Chunyu Chen;Xinsheng Wu;An Chen 2020 International Symposium on Autonomous Systems (ISAS)
[34] Lu Dongdong;Tie Wenjie;Lei Songlin;Qiu Xiaolan 2021 CIE International Conference on Radar (Radar)
[35] Azarpour, M., Seyyedsalehi, S. A., & Taherkhani, A. (2010). Robust pattern recognition using chaotic dynamics in Attractor Recurrent Neural Network. The 2010 International Joint Conference on Neural Networks (IJCNN). doi:10.1109/ijcnn.2010.5596375
[36] Samarawickrama, A. J. P., & Fernando, T. G. I. (2017). A recurrent neural network approach in predicting daily stock prices an application to the Sri Lankan stock market. 2017 IEEE International Conference on Industrial and Information Systems (ICIIS). doi:10.1109/iciinfs.2017.8300345
[37] Chernigovskiy, A. S., Tsarev, R. Y., & Knyazkov, A. N. (2015). Hu's algorithm application for task scheduling in N-version software for satellite communications control systems. 2015 International Siberian Conference on Control and Communications (SIBCON).
[38] Cong, S., Yu, M., & Dai, Y. (2010). Approximation performance analysis of recurrent neural networks. 2010 Sixth International Conference on Natural Computation.
[39] Pakanzad, S. N., & Monkaresi, H. (2020). Providing a Hybrid Approach for Detecting Malicious Traffic on the Computer Networks Using Convolutional Neural Networks. 2020 28th Iranian Conference on Electrical Engineering (ICEE).
[40] Azimirad, V., Mortazavy, V., & Sani, M. F. (2017). Real-time optimal trajectory planning of mobile robot in presence of obstacle through Generalized Regression Neural Network. 2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI).
[41] Gope, J., Chowdhury, S., Panda, M., Bhattacharyya, N., & Bhadra, S. (2016). JMS hybrid single electron model and its simulation using MATLAB. 2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON).
[42] Tan, Y., Chen, Y., Li, Y., & Cao, Y. (2020). Linearizing Power Flow Model: A Hybrid Physical Model-driven and Data-driven Approach. IEEE Transactions on Power Systems, 1–1. doi:10.1109/tpwrs.2020.2975455
[43] Bishop, J., Burgess, J., Ramos, C., Driggs, J. B., Williams, T., Tossell, C. C., … Visser, E. J. de. (2020). CHAOPT: A Testbed for Evaluating Human-Autonomy Team Collaboration Using the Video Game Overcooked!2.2020
[44] Gope, J., Chowdhury, S., Panda, M., Bhattacharyya, N., & Bhadra, S. (2016). JMS hybrid single electron model and its simulation using MATLAB. 2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON).
[45] Tan, Y., Chen, Y., Li, Y., & Cao, Y. (2020). Linearizing Power Flow Model: A Hybrid Physical Model-driven and Data-driven Approach. IEEE Transactions on Power Systems, 1–1. doi:10.1109/tpwrs.2020.2975455
[46] Bishop, J., Burgess, J., Ramos, C., Driggs, J. B., Williams, T., Tossell, C. C., … Visser, E. J. de. (2020). CHAOPT: A Testbed for Evaluating Human-Autonomy Team Collaboration Using the Video Game Overcooked!2.2020
[47] Hazrati, O., Ghaffarzadegan, S., & Hansen, J. H. L. (2015). Leveraging automatic speech recognition in cochlear implants for improved speech intelligibility under reverberation. 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). doi:10.1109/icassp.2015.7178941
[48] Shen, W., Yu, R. P., Seide, F., & Wu, J. (2009). Automatic punctuation generation for speech. 2009 IEEE Workshop on Automatic Speech Recognition & Understanding. doi:10.1109/asru.2009.5373365
[49] Prodeus, A., & Kukharicheva, K. (2016). Training of automatic speech recognition system on noised speech. 2016 4th International Conference on Methods and Systems of Navigation and Motion Control (MSNMC). doi:10.1109/msnmc.2016.7783147
[50] Martinez-Hinarejos, C.-D., Granell-Romero, E., & Romero-Gomez, V. (2018). Comparing Different Feedback Modalities in Assisted Transcription of Manuscripts. 2018 13th IAPR International Workshop on Document Analysis Systems (DAS). doi:10.1109/das.2018.13
[51] Chaloupka, J., Palecek, K., & Cerva, P. (2021). Audio-visual Broadcast Transcription System Using Artificial Neural Networks. 2021 IEEE International Workshop of Electronics, Control, Measurement, Signals and Their Application to Mechatronics (ECMSM). doi:10.1109/ecmsm51310.2021.94688
[52] Ramabhadran, B., Jing Huang, & Picheny, M. (n.d.). Towards automatic transcription of large spoken archives - English ASR for the MALACH project. 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). doi:10.1109/icassp.2003.1198756



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)