**ReadIt: A Multilingual Text to Speech Tool for OCR and Text Summarization Models**

Srinidhi Shukla, Nithin Teja Reddy Gottam, Sandeep Srinivas Guthula, Divyaj Reddy Nalla

*University of North Texas*

**Abstract.** We talk about two processes in this paper- Text extraction from Image and converting it to speech and abstractive summarization, where the first one aims for the extraction of text present in the image to converting it into a speech and the later one aims for the process of constructing semantically relevant shorter sentences called as summaries by capturing the essence of the overall meaning of the source text. The image-text-speech feature has been restricted to four languages, and the whole process will be executed by a front-end tool that requires you to simply upload an image to listen the text and a document to get a summary. This image-text-speech is aimed at people who would like to listen to the text present in the image rather than read it, particularly blind people. We use the Deep learning models and libraries like NLP and Networkx and we limit the image-text-speech implementation to four languages: Telugu, Kannada, Hindi, and English, and Abstractive text summarization to just one language: English. The proposed architecture is built on Tesseract OCR – Optical Character Recognition, which is available under the Apache 2.0 license, and the GTT's – Google Text to Speech – API for the front-end tool, as well as Tkinter. The model was validated with a variety of images to listen to the text and by manually uploading large documents to create a two-line sentence description of the source text, and it produced positive qualitative results. The main focus was to address the multilingual limitation of NLP.

**Keywords:** Text to Speech, OCR, Multilingual, NLP, Summarization, Image-text-speech, Tesseract, TextRank algorithm.

1. **Introduction**

Textual data is booming in today's world, and because it's not practical for everyone to read the entire thing and comprehend it, what we need is a method to condense the text while maintaining the essence of the source text. Even, in the case of visually impaired persons and those with color blindness who are unable to interpret the text present in an image, using a method such as ReadIt tool to import the image and extract the text present in the image and read it out will be beneficial.

**1.1 Image-Text-Speech Conversion**

285 million people on our planet of 7.4 billion people are visually disabled, with 39 million being fully blind (no vision at all) and 246 million having moderate to extreme visual disability (WHO, 2011). It is estimated that by 2020, there will be 75 million blind people and 200 million people with visual disability.[5]. Since reading is so important in people's everyday lives (text can be seen in newspapers, commercial products, signboards, and digital displays, among other places), visually disabled people have a hard time reading them. We used optical character recognition (OCR) to enable them to listen to the text written on the image.

Optical Character Recognition (OCR) is a technique for extracting text from images and translating it to an electronic format. Handwritten text, as well as typed text such as documents, receipts, and name cards, printed on the images. It has two parts: the first is text detection, which determines the textual part of the image and imports the localization of the text within the image for the second section of OCR, text recognition, which extracts the text from the image. The extracted text is then translated into an audio file (mp3) using GTT's Google Text to Speech API.

A picture containing text, sign, stop, tree

Description automatically generated

Fig 1: Image-Text-Speech

* 1. **Text Summarization and Speech Conversion**

Text summarization is the process of producing a concise, factual, and semantically meaningful interpretation of a text. Extractive Text Summarization and Abstractive Text Summarization are the two types of text summarization.

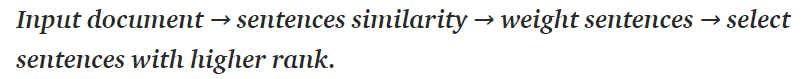
Diagram

Description automatically generated

Fig 2: Text Summarization

The abstractive text summarization approach generates new semantically meaningful sentences and can also generate summaries by rephrasing or using terms not used in the source text. These approaches are usually more difficult since, in order to provide precise and semantically correct summaries, the model must first comprehend the context of the text before attempting to articulate the interpretation with the appropriate phrases and vocabulary.

The proposed Extractive text summarization methods create summaries by copying portions of the source text, assigning a value to the terms, joining those sentences to form a summary, and then reading the summary out loud using GTT's API. With NLP and Python, this method weights the most relevant parts of sentences and uses them to construct the description.



We used an unsupervised learning technique to find sentence similarities and rating here for flexibility, since there is no need to train the model with a large amount of text beforehand. The model then converts the text into Speech using GTT's Google Text to Speech API.

1. **Related Work**

We wanted to make a front-end interface that would enable people to work with image-text-speech and text summarization with speech. A few similar works from which we drew our inspiration are mentioned below.

1. Image Processing Based Scene-Text Detection and Recognition with Tesseract by Ebin Zacharias, Martin Teuchler and Bénédicte Bernier. The aim of this project is to detect and recognize words in natural photographs. The intended challenge is considerably more difficult than translating text from scanned papers. Because of the limited number of photographs available, the use case in focus makes it easier to detect the text field in natural scenes with greater precision.
2. Smart Summarizer For Blind People by Mona Teja K , Mohan Sai.S, H S S S Raviteja ,Sai Kushagra. This paper discusses an effective technique for condensing news into relevant keywords in order to avoid having to read the whole text each time. Many APIs and modules, such as tesseract and GTTS, are explored and introduced in depth in this article, as are many algorithms, such as Luhn's Algorithm, Latent Semantic Analysis Algorithm, and Text Ranking Algorithm.

The aim was to combine this into one platform so that everyone, including colorblind individuals, could hear the text in the picture and extract audio from the summarized text along with addressing the limitation of multilingual processing.

1. **Architecture**

Graphical user interface

Description automatically generated with low confidence

Fig 3: Architecture of ReadIt tool

**4. Proposed Model**

**4.1. Image-text-speech Conversion Model**

The system's module is the TESSERACT GOOGLE API, which is an optical character recognition (OCR) engine that supports Unicode and can identify several languages out of the box. It is adaptable and can be learned to recognize other languages as well; however, for the purposes of this paper, we have restricted the model to four languages: Hindi, Kannada, Telugu, and English. The Tesseract can detect over 100 languages, and Google has boosted it with a neural network framework based on LSTM starting with version 4. (long short-term memory).

Timeline

Description automatically generated

Fig 4: OCR Process flow

To function as efficiently as possible, OCR is typically split into many sub-processes. The following are the sub-processes:

* Preprocessing of the Image
* Text Localization
* Character Segmentation
* Character Recognition
* Post Processing

Chart, bubble chart

Description automatically generated

Fig 5: OCR Workflow

The following are the Libraries that are used in the model:

1. Pytesseract is a library that acts as a wrapper for Tesseract in Python. Tesseract makes use of Leptonica, a pedagogically based open source site with tools that can be used for a wide range of image recognition and image analysis applications.
2. PIL is a Python imaging library that allows you to open, manipulate, and save various image file formats. The image library's core is designed to provide quick access to data stored in a few simple pixel formats. It should serve as a strong foundation for an image processing tool in general.
3. Nltk - Natural language toolkit for data preprocessing and text related work.
4. NumPy – used for working with arrays.
5. OS – It is used for interacting with operating system.
6. GTT’s – Google Text-to-Speech is a Python library and command-line utility that allows you to interact with the Google Translate text-to-speech API.

**4.2.** **Text Summarization and Speech Conversion Model**

TextRankis a graph-based text processing ranking model that can be used to find the most relevant sentences in a text as well as keywords.

Chart

Description automatically generated

Fig 6: TextRank Algorithm Process Flow

### **Relevant sentences identification:**

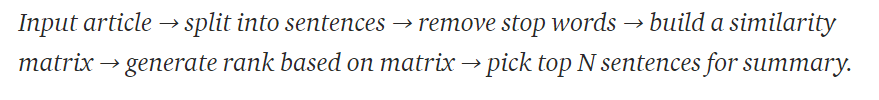
To find the most relevant sentences in text, a graph is created, with the vertices representing each sentence in the document and the edges between sentences based on content overlap, i.e. the amount of terms that two sentences share.

The sentences are fed into the PageRank algorithm, which selects the most relevant sentences, based on this network of sentences. We can now remove only the most relevant sentences from the text while extracting a summary.

### **Relevant keywords identification**

The text rank algorithm creates a word network to identify appropriate keywords. This network is built by examining which terms are connected to one another. If two words accompany each other in the text, a link is formed between them; the link is given more weight if the two words appear more often next to each other in the text.

The PageRank algorithm is applied to the resulting network to determine the importance of each expression. The top third of all of these terms is retained and considered acceptable. Following that, a keywords table is created by grouping the related terms together if they appear in the text after one another.



The following are the libraries used:

1. Nltk: NLTK is a popular Python framework for dealing with human language information. It includes a collection of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, as well as wrappers for industrial-strength NLP libraries and an active discussion site, as well as easy-to-use interfaces to over 50 corpora and lexical tools like WordNet.
2. NetworkX: In Python, NetworkX stands for network analysis. It's primarily used to create, manipulate, and analyze complex graphs.
3. Langdetect: The concept behind language detection is to find a character in the expressions and words in the text. The key idea is to find widely found English words like to and of. The Langdetect library is a 55-language port of Google's language-detection library.
4. Json: It is a built-in python package used in working with Json files.
5. Io: Handles the file related input and output operations.
6. Time: It provides time in code, such as waiting time during code execution and calculating the code's performance.

**4.3 The Front-end Execution:**

In the final stage, we use Tkinter to create a GUI that incorporates all of these models and makes it simpler for the user. While Python has several GUI frameworks, Tkinter is the only one that is used in the standard library. Tkinter has a number of advantages. The code is cross-platform, meaning it runs on Windows, Mac OS X, and Linux. Tkinter programs appear like they belong to the platform they're running on because visual elements are made using native operating system elements.

Tkinter makes it easy to make a graphical user interface using the steps below. −

* Import the Tkinter module to the project.
* Build the main window for the GUI application.
* In the GUI framework, add one or more of the above-mentioned widgets.
* Enter the main event loop to respond to each user-triggered event.

1. **Training:**

Tesseract has three collections of official trained data files for Tesseract versions 4.00 and up that were trained at Google. These can be found in three different libraries- ‘tessdata\_fast’ (Sep 2017) Integer models have the highest “value for money” in terms of speed and accuracy. ‘tessdata\_ best’ (Sept 2017) produces the best performance on Google's eval data but is slower and uses Float models. Only these templates can be used as a foundation for finetuning preparation. ‘tessdata’. (Nov 2016 and Sep 2017) There are 2016 tesseract versions that haven't been modified. Integer versions of the tessdata best LSTM variants have been added to the LSTM models. (Legacy tesseract models based on cubes for Hindi, Arabic, and other languages have been removed.) The TextRank keyword extraction algorithm is fully unsupervised and works like this: To allow the application of syntactic filters, the text is first tokenized and annotated with part of speech tags – a preprocessing step. So we provide images with white or background with minimal noice as an input to the OCR model and text documents for the Text summarization with speech model.

1. **Results and Evaluation:**

The output of the code is shown below. It shows a front-end interface with two choices for the user to choose from:

Graphical user interface, text, application, email

Description automatically generated

Fig 7: ReadIt Front-end Display

* **Execution of Image-Text-Speech model:**

Step 1: Uploading the images and converting of the text present on the image and displaying it

Image 1:

A picture containing background pattern

Description automatically generated

Fig 8: Input Image 1

Output:

Graphical user interface, text

Description automatically generated

Fig 9: Output 1

Image 2:

Company name

Description automatically generated

Fig 10: Input Image 2

Output:

Graphical user interface, text, application

Description automatically generated

Fig 11: Output

Image 3:

Text, letter

Description automatically generated

Fig 12: Input Image 3

Output:

Text

Description automatically generated with medium confidence

Fig 13: Output

Step 2: Saving the converted text into audio file to play.

A screenshot of a computer screen

Description automatically generated with medium confidence

Fig 14: Screenshot of Saved Audio File

* **Execution of the Text summarization and speech conversion model:**

Step 1: Uploading the document and then getting the summarized text:

Graphical user interface, text

Description automatically generated

Fig 15: Uploading the Document File

Graphical user interface, text, application, Word

Description automatically generated

Fig 16: JSON Input Text Document File

Text

Description automatically generated

Fig 17: Text Summarization Output

Step 2: Storing the file into audio Mp3 format to play it out:

A screenshot of a computer screen

Description automatically generated with medium confidence

Fig 18: Screenshot of Saved Audio File

We only limited the summary to two lines, but the number of lines may be increased if necessary.

1. **Conclusion**

We were successful in developing a front-end interface that allows us to incorporate Image-text-speech and text summarization and speech conversion depending on the user's request. We restricted the languages in the image-text-speech model to four, and the model was accurate in producing text when images included the listed languages. The extractive approach is followed by the text summarization model, which provides the summary in two lines since we have narrowed it to that. The best aspect is the conversion of the summarized text into an audio Mp3 format. Because of the libraries we used, the whole model does not require any data to be trained, and the results can be obtained simply by uploading an image or file and making a selection on the front-end interface. This concept was mostly aimed at making it simple for everyone, even blind people, to convert text or image data to audio, and we plan to add more features in the future to make it more practical and reliable. For fast conversions and convenient access for the disabled, this tool may also be used as a plug-in or add-on for applications or smart devices (such as a smart watch or Alexa).

1. **References**

[1]Zacharias, E., Teuchler, M., and Bernier, B., “Image Processing Based Scene-Text Detection and Recognition with Tesseract”, 2020.

[2]Teja K, M., Sai. S, M., Raviteja D, H. S. S. S., and V, S. K. P., “Smart Summarizer for Blind People”, 2020.

[3]Bharath B, M., Gowtham B, A., and M, A., “Neural Abstractive Text Summarizer for Telugu Language”, 2021.

[4]Galatolo, F. A., Cimino, M. G. C. A., and Vaglini, G., “Generating images from caption and vice versa via CLIP-Guided Generative Latent Space Search”, 2021.