A Helping Hand for The Visually Challenged, Mute and Deaf people

S A Hariprasad[1], Kishore Kumar N[2], Kunal Agarwal[3], S Ramkrithik[4], Pardhasaradhi Tirumalasetti[5], A Aniruddha[6]

[1,2,3,4,5,6]*Students, Dept. of Computer Science and Engineering (SCOPE), VIT University, Tamil Nadu, India*

---***---

Abstract: *Daily activities including Communication, Navigation, etc. are certainly very challenging tasks for visually impaired, blind and deaf-mute people. Indoor navigation itself is certainly becoming a harder task for blind, visually impaired people and dead-mute people. As far as it is observed for the non-visually impaired, it is even worse for the visually impaired. People with visual disabilities or let's say blinds are often depending upon the external assistance like trained dogs, humans, or special devices as support systems for making decisions. Hence blind people need an assistive device that will allow blind user to navigate freely and this requirement has become crucial. So, our all-in-one application is implemented to make the lives of challenged people easier.*

To ease the communication for the visually blind, we have implemented a language in which they are comfortable, which is Braille. Our project simply will convert the given text into braille so they can easily understand and reply too using the vice versa feature.

We have used speech recognition to recognize and coordinate with the hand signals. Detection is done by python libraries which use voice and speech recognition to identify what the mute person is conveying. For speech recognition NumPy, matplotlib.pyplot, matplotlib.pyplot.cv2 libraries have been deployed.

Key Words: **Image processing, Braille Code, Sign Language, Speech recognition, Natural Language Processing.**

1. INTRODUCTION

With progresses in new technologies, cell phones have filled in notoriety to become perhaps the most well-known customer device. Phones are vital piece of current life. Large numbers of us need to settle on a decision or communicate something specific at whenever from anyplace.

The visually impaired individuals face difficulties day by day in speaking with their general surroundings. They need to rely upon their located associates for settling on a telephone decision and getting to other portable functionalities. This framework is a voice perceiving application for cell phones that permit admittance to the vast majority of the functionalities of the telephone and will make it feasible for visually hindered individuals to associate with the general public. The located client's kin with restricted perusing capacity can likewise utilize our website/application who are knowledgeable with the Braille language. Additionally, our point is to work on the correspondence with individuals who has hearing hardships and utilizing any sign language to articulate their thoughts. At the principal sight, as a thought, how troublesome could make a sign dialects converter. We'll catch capacities and specialized provisions to the motion catch of sign to voice Change.

1.1. Background Study

1) Research on the Hand Gesture Recognition Based on Deep Learning:

Study: With the quick improvement of PC vision, the interest for communication among human and machine is turning out to be increasingly broad. Since hand signals can communicate advanced data, the hand motion acknowledgment is generally utilized in robot control, shrewd furnishings and different viewpoints. The paper understands the division of hand signals by building up the skin shading model and AdaBoost classifier dependent on haar as indicated by the distinction of skin tone for hand motions, just as the denaturation of hand motions with one edge of video being cut for investigation. In such manner, the human hand is divided from the convoluted foundation, the continuous hand motion following is additionally acknowledged by Cam Shift calculation. Directly generated by picking lines from the corpus and giving it as questions.

2) End-to-End Attention-based Large Vocabulary Speech Recognition:

Study: The framework proposes here is a neural organization that can plan groupings of discourse edges to arrangements of characters. While the entire framework is differentiable and can be prepared straightforwardly to play out the main job, it can in any case be separated into various useful parts that cooperate to figure out how to encode the discourse signal into an appropriate component portrayal and to unravel this portrayal into a succession of characters.

3) SQuAD: 100,000+ Questions for Machine Comprehension of Text:

Study: Candidate answer age. For each of the four strategies, as opposed to thinking about all O(L2) ranges as competitor replies, where L is the quantity of words in the sentence, we just use traverses which are constituents in the voting demographic parse created by Stanford CoreNLP. Disregarding accentuation and articles, we track down that 77.3% of the right replies in the improvement set are constituents. This places a successful roof on the exactness of our strategies. During preparing, when the right reply of a model is certainly not a constituent, we utilize the briefest constituent containing the right reply as the objective.

4) Adaptive Document Retrieval for Deep Question Answering

Study: Threshold-Based Retrieval As a credulous pattern, we propose a basic edge-based heuristic. That is, not set in stone with the end goal that the total certainty score arrives at a decent edge. Ordinal Regression-further carry out a teachable classifier as an ordinal edge relapse which is custom fitted to positioning assignments. It is additionally expected that total certainty liable to be direct. The classifier then, at that point approximates ni with an expectation yi that signifies the situation of the primary important report containing the ideal reply.

5) Attention Is All You Need

Study: Most cutthroat neural grouping transduction models have an encoder-decoder structure. Here, the encoder maps an info grouping of image portrayals to an arrangement of nonstop portrayals z = (z1; :::; zn). Given z, the decoder then, at that point produces a yield succession (y1; :::; ym) of images each component in turn. At each progression the model is auto-backward, devouring the recently created images as extra information while producing the following. The Transformer follows this general engineering utilizing stacked self-consideration and point-wise, completely associated layers for both the encoder and decoder

2. ARCHITECTURE DIAGRAM

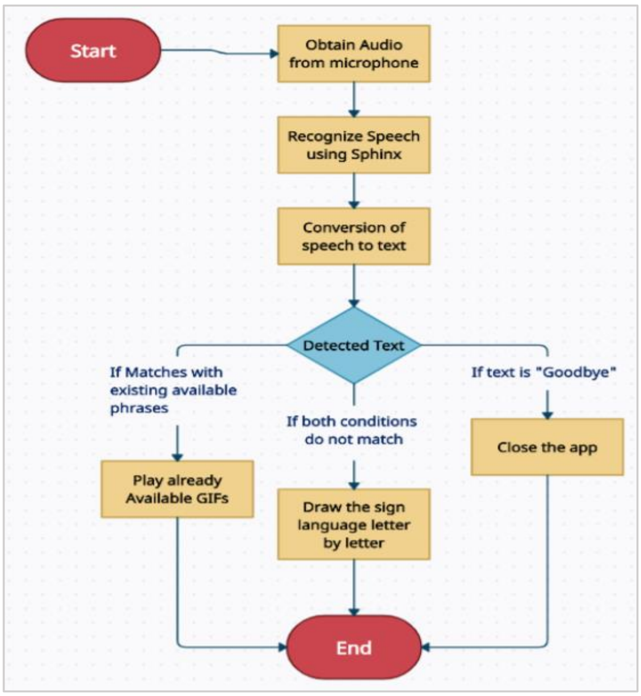

Start
Obtain Audio from microphone
Recognize Speech using Sphinx
Conversion of speech to text
Detected Text
If Matches with existing available phrases
If both conditions do not match
If text is "Goodbye"
Play already Available GIFs
Draw the sign language letter by letter
Close the app
End

Fig -1: Sign Language Converter

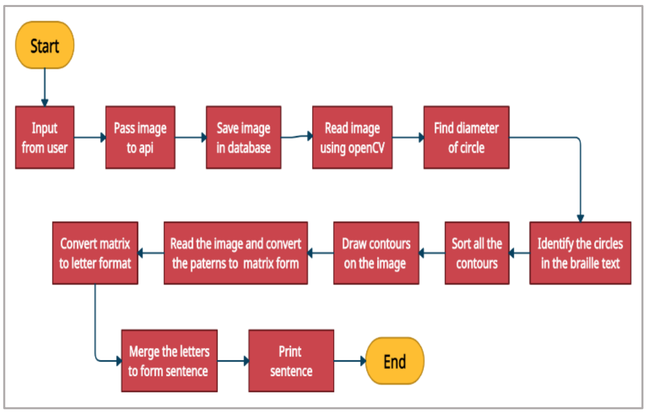

Start
Input from user
Pass image to api
Save image in database
Read image using openCV
Find diameter of circle
Identify the circles in the braille text
Sort all the contours
Draw contours on the image
Read the image and convert the paterns to matrix form
Convert matrix to letter format
Merge the letters to form sentence
Print sentence
End

Fig -2: Braille image-text Converter

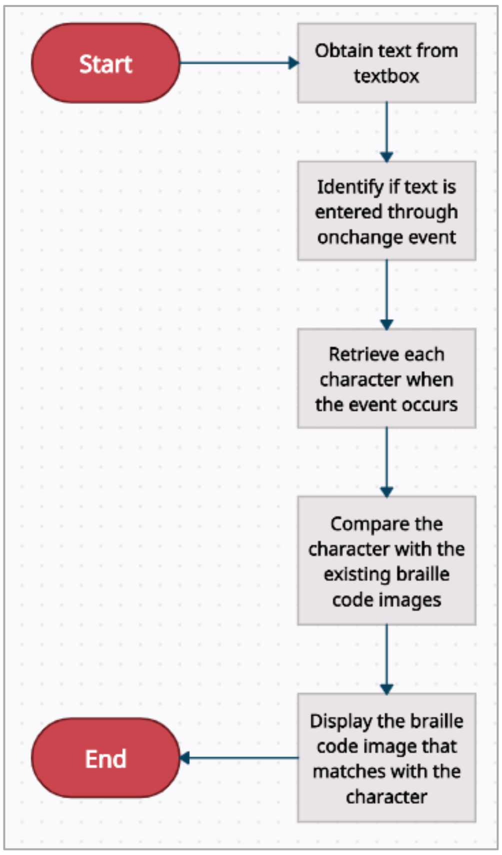

Start
Obtain text from textbox
Identify if text is entered through onchange event
Retrieve each character when the event occurs
Compare the character with the existing braille code images
Display the braille code image that matches with the character
End

Figure 3: Text-braille code Converter

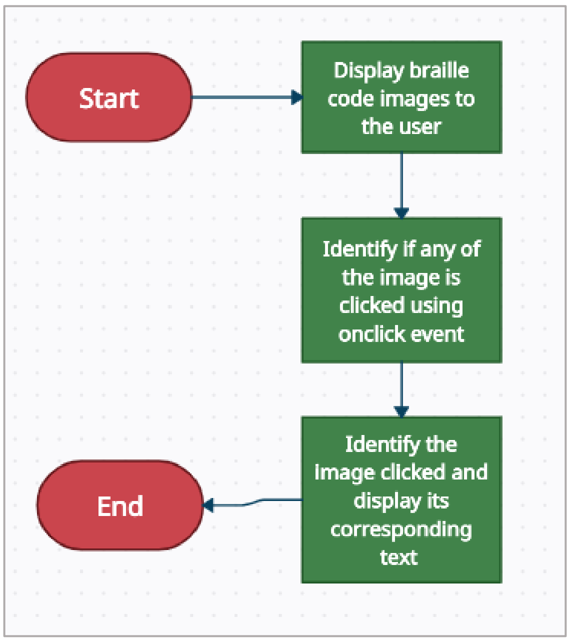

Start
Display braille code images to the user
Identify if any of the image is clicked using onclick event
Identify the Image clicked and display its corresponding text
End

Figure 4: Braille keyboard

3. METHODOLOGY AND PROPOSED MODEL

Sign Language Converter:

The language used for communication with deaf people is sign language. This is like their native language; in other words, it is like a mother tongue. As deaf people can't hear our voices in order to communicate with them, we use signs or symbols. But everyone may not know sign language as it is not the necessary one to be learned. so, to overcome this problem we proposed an application that takes live or recorded voice as input and communicates with the deaf through signs or symbols. In order, to make this application more interactive we have trained the data sets with images and also small GIFs. This was implemented using Python. For our implementation we used libraries like speech recognition, NumPy. matplotlib.pyplot, matplotlib.pyplot.cv2 etc. Now, when the application is launched, we get there two option to begin our application like live recording and all done. All done is to close the application and the live recording is to get the input voice from the user through the microphone which is done using recognizer function which is python inbuilt library. So store the recorded voice in temporary memory and we do text pre-processing using natural language processing techniques and if the voice is not able to be recorded we display a message called "could not listen" generally this is a user choice of message and for the text detected we perform dictionary based Machine translation that is to search the words in our trained dictionary if they were not found just spell the word using sign language which we have already trained from the alphabets A-Z. Here to make our job easier and make the application more interactive we have added GIFs and some small video clips which are generally based on daily routines like "Hello"," Good Morning" ,"what is your name" etc. so that for these common questions and compliments our job gets done easier and here this is the main purpose of this application ,but for every application there must be a termination phase also so to achieve this we made a terminating command called "goodbye" so when the application receives this command during inputting audio phase it just get terminated.

1. Get the input voice from the user through the microphone.
    1.1. Listen for a particular amount of time.
    1.2. Listened voice gets converted into text through speech recognition libraries.
2. After listening the voice is captured and saved in the temporary memory.
3. Through the temporary memory the voice is fetched and converted to text.
    3.1. To proceed with further manipulation, convert

the entire string of text to lowercase letters

4. So, after getting the text each and every character of the inputted text is searched throughout the data set.
    4.1. If the text is "goodbye" then the application exits, as this is the existing command.
    4.2. And if the detected text is not "goodbye" then it first searches the word in the predefined dictionary images and GIFs.
    4.3. If it is not found there, then spells the word using symbols with some delay in the image display actions.
    4.4. Loops from step 2 till the speech ends
5. If an error occurs in Step 1, display "Could not listen."

Text – Braille code Converter:

This is implemented using React-JS. When the application is launched the user will be given a text box to enter the text. Each text is converted to its lowercase and it is searched among the database for a match with our saved braille code images. The image is saved as "letter.png" form. So, when the user enters a text, the application searches for the text in the database and displays that image to the user. This is dynamic application as the output is displayed based on the text currently in the textbox and it is identified using the onchange event in the DOM

1. Get the input from the user through the textbox.
2. Identify if any change occurred in the textbox
3. Retrieve each character of the text from the textbox.
    3.1. Search the character with the images available in the database.
    3.2. If the character matches with the image name, display the image to the user.
    3.3. Loop from step 2 till the complete text is parsed.

Braille image – Text Converter:

The language used by blind people to communicate with others. These are their native alphabets used for forming sentence. As blind people cannot see or read the words and sentence, they use these symbols to communicate with others and also write sentence. But it's not very easy to read and understand all documents written in braille form hence we have developed an application using image processing to convert braille image to text so that people can read the braille text without understanding each alphabet and also help in communication between blind people and others. This application is developed using python and libraries such as OpenCV, skimage and many more. We have the option of uploading the image and using react framework and process the work using flask API. The process starts with uploading the image in react and passing the image to flask api using axiom library. After passing the image to flask API the image is processed and converted to OpenCV format for processing. After processing the image, the diameter and centers of circles are found and converted to matrix form. After conversion the contours are drawn and lines are drawn between letters for differentiating. After separation based on matrix value the alphabets are mapped and merged. After the sentence is formed it is passed back for printing in react app.

1. Get the input file from the user in image format
    1.1. If no file uploaded message file not uploaded, please upload file
2. After uploading the image pass the image to the API
3. After uploading image save the image in the database
4. Read the image using OpenCV
    4.1. Convert image to black and white form
    4.2. Blur image to remove noise
    4.3. Get image edges
    4.4. Getting contours for image
    4.5. Group contours with a minimum of four points
    4.6. Apply Otus thresholding method to binaries the image
    4.7. Erode and dilate image to remove unnecessary noise
    4.8. Return image details
5. Find the diameter of the circle
6. Find the circles of the braille text
7. Sort all the contours
8. Draw contours
9. Convert contour to matrix form of letters

10. Get all the letters

11. Group letters and print output and stop process the process

Braille Keyboard:

This is implemented using React-JS. When the application is launched, a braille code keyboard will be displayed to the user. The keyboard is set of braille code images which is displayed to the user from the database. When the user presses any braille code image, the event is captured using onclick event and the corresponding text for that braille code will be displayed. This is a dynamic application in which we can enter the modify the text inside the textbox and also convert the braille code to text and any click in the keyboard is captured through onclick event.

1. The braille code images will be displayed to the user.

2. Identify if the user clicks any image using the onclick event

3. Identify the image and display its corresponding text to the user in the textbox

   3.1. Loop from step 2.

4. OUTPUTS

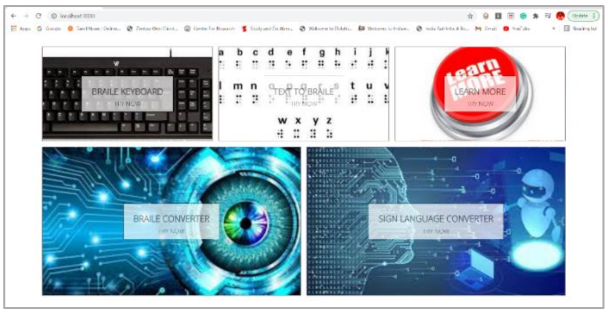
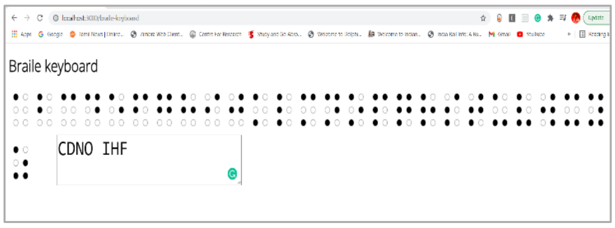


Fig -5: Homepage


Braile keyboard

CDNO IHF

Fig -6: Braille Keyboard Output

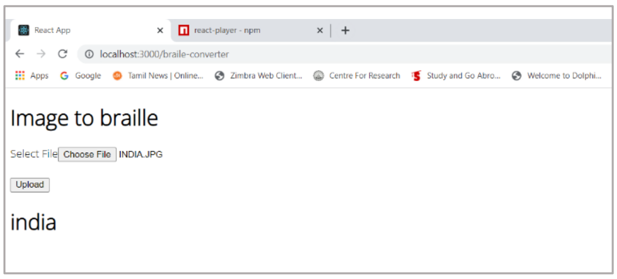


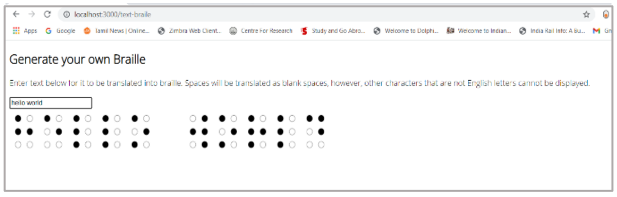
Image to braille

Select File Choose File INDIA.JPG

Upload

india

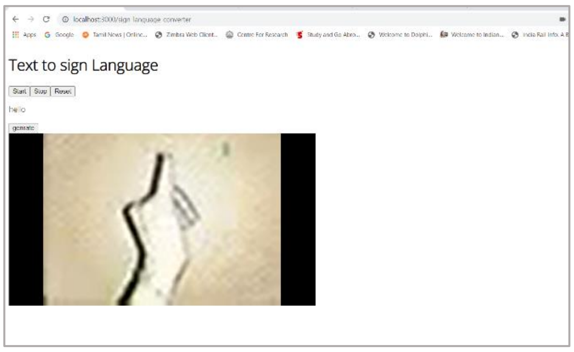
Fig -7: Braille image - text Output


Generate your own Braille

Enter text below for it to be translated into braille. Spaces will be translated as blank spaces, however, other characters that are not English letters cannot be displayed.

hello world

Fig -8: Text – braille code Output


Text to sign Language

Start Stop Reset

hello

generate

Fig -9: Speech – Sign language Output

5. CONCLUSIONS

So, in this project, with the skills and the motivation we had to do something for the community, we made this application which will help the blind, deaf and mute people a way to communicate better. Users will be able to use the features of this application according to their needs. Our helping hand is an application which conducts a conversation by means of voice recognition and hand gestures. Such projects are regularly intended to convincingly reproduce to assist people with various assignments. This paper presents speech recognition and incorporation of gesture-based communication through python libraries with assistance and gives us hand gestures visuals.

Using React we made the application more user friendly and easy to use and the libraries in python helped us to enhance it. We have built this project from scratch so that the users can easily access our tools and use the system to its fullest capabilities.

REFERENCES

[1] Bahdanau, Dzmitry, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. "End-to-end attention-based large vocabulary speech recognition." In 2016 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp. 4945-4949. IEEE, 2016.

[2] Chen, Danqi, Adam Fisch, Jason Weston, and Antoine Bordes. "Reading wikipedia to answer open-domain questions." arXiv preprint arXiv:1704.00051 (2017).

[3] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

[4] Hermann, Karl Moritz, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. "Teaching machines to read and comprehend." Advances in neural information processing systems 28 (2015): 1693-1701.

[5] Karpukhin, Vladimir, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. "Dense passage retrieval for open-domain question answering." arXiv preprint arXiv:2004.04906 (2020).

[6] Kratzwald, Bernhard, and Stefan Feuerriegel. "Adaptive document retrieval for deep question answering." arXiv preprint arXiv:1808.06528 (2018).

[7] Kupiec, Julian. "MURAX: A robust linguistic approach for question answering using an on-line encyclopedia." In Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 181-190. 1993.

[8] Lan, Zhenzhong, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. "Albert: A lite bert for self-supervised learning of language representations." arXiv preprint arXiv:1909.11942 (2019).

[9] Lee, Jinhyuk, Seongjun Yun, Hyunjae Kim, Miyoung Ko, and Jaewoo Kang. "Ranking paragraphs for improving answer recall in open-domain question answering." arXiv preprint arXiv:1810.00494 (2018).

[10] Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. "Roberta: A robustly optimized bert pretraining approach." arXiv preprint arXiv:1907.11692 (2019).

[11] Moldovan, Dan, Sanda Harabagiu, Marius Pasca, Rada Mihalcea, Roxana Girju, Richard Goodrum, and Vasile Rus. "The structure and performance of an open-domain question answering system." In Proceedings of the 38th annual meeting of the Association for Computational Linguistics, pp. 563-570. 2000.

[12] Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep contextualized word representations." arXiv preprint arXiv:1802.05365 (2018).

[13] Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. "Improving language understanding by generative pre-training." (2018).

[14] Rajpurkar, Pranav, Jian Zhang, Konstantin Lopyrev, and Percy Liang. "Squad: 100,000+ questions for machine comprehension of text." arXiv preprint arXiv:1606.05250 (2016).

[15] Reddy, Siva, Danqi Chen, and Christopher D. Manning. "Coqa: A conversational question answering challenge." Transactions of the Association for Computational Linguistics 7 (2019): 249-266.

[16] Simmons, Robert F., Sheldon Klein, and Keren McConlogue. "Indexing and dependency logic for answering English questions." American Documentation 15, no. 3 (1964): 196-204.

[17] Shridhar, A. "A beginner's guide to deep learning." (2017).

[18] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In Advances in neural information processing systems, pp.5998-6008. 2017.

[19] Wang, Wenhui, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. "Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers." arXiv preprint arXiv:2002.10957 (2020).

[20] Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac et al. "Huggingface's transformers: State-of-the-art natural language processing." arXiv preprint arXiv:1910.03771 (2019).

[21] Yan, Zhao, Nan Duan, Junwei Bao, Peng Chen, Ming Zhou, Zhoujun Li, and Jianshe Zhou. "Docchat: An information retrieval approach for chatbot engines using unstructured documents." In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 516-525. 2016.

[22] Yang, Yi, Wen-tau Yih, and Christopher Meek. "Wikiqa: A challenge dataset for open-domain question answering." In Proceedings of the 2015 conference on empirical methods in natural language processing, pp.2013-2018. 2015.