Picolo: A Smart Assistant

Tanvi Aggarwal
ID: 114353100
Stony Brook, NY
taggarwal@cs.stonybrook.edu

ABSTRACT

Nowadays, a majority of meetings and classes are being conducted online through video conferencing platforms. Whenever the presenter wants to share information such as links or other data, they will most likely copy and paste the text in the meeting chat, or a separate document which needs to be shared with the attendees. This creates an overhead for the host and is inconvenient for the attendees. In another similar use case, sometimes we need to copy text information from an image, and the only option is to manually re-type the text, which can be time-consuming and error prone. To improve these interaction experiences, we introduce a smart assistant: Picolo, with two design interfaces PicoloExtension and PicoloDesk that leverage OCR (Optical Character Recognition) to help extract textual information from images, in real time. We provide a comprehensive evaluation of these interfaces against the manual entry interface (and each other), using discount usability testing and user testing. We analyze the results of descriptive and inferential statistics on the evaluation study, proving significant performance gains using the proposed PicoloExtension. Additionally, we discuss enhancements for future work.

Author Keywords

Optical Character Recognition; OCR; Chrome extensions; Picolo; Productivity; Data extraction.

INTRODUCTION

Technology is advancing rapidly and so is the complexity of problems we aim to solve. Human computer interaction is constantly evolving, directly impacting the things we do on a day-to-day basis and even as our lives get busier, we also have more tools than ever, to boost our productivity and increase the scale of operations. As the COVID-19 pandemic continues into its third year, it has perhaps led to several changes that will have a long-lasting impact on our lives, for better or for worse.

One such shift was that almost overnight, our world went online, as a majority of meetings and classes started getting conducted through video conferencing platforms. This "hybrid" mode as it is often called, is the new normal and it is here to stay. One of the major challenges that it has brought along with it is this reduced interpersonal communication and its psychological impact is being studied. It also brings in a sudden change in our work environment, which means that we need new tools to help us adapt to the online style of meetings and classes.

In this study, we explore one such application and propose an interface for a Smart Assistant: Picolo. In a video conferencing meeting, whenever the presenter wants to share information such as links or other data, they will most likely copy and paste the text in the meeting chat, or a separate document which needs to be shared with the attendees. This creates an overhead for the host and is inconvenient for the attendees. In another similar use case, sometimes we need to copy text information from an image, and the only option is to manually re-type the text, which can be time-consuming and error prone. Picolo is an interface to improve these interaction experiences that leverages Optical Character Recognition (referred to as OCR) using Tesseract [8] to extract textual information from the images.

The contributions in this paper include two interfaces: PicoloExtension, a web-browser extension to allow the user to take a screenshot of the part of the screen they want to extract text from, and PicoloDesk, a desktop application in the form of a script to perform the same.

RELATED WORK

Optical Character Recognition (OCR) is one of the earliest areas of artificial intelligence research. OCR is a specialized technology to perceive the characters of a text within the images like printed books, photos, or scanned documents. It converts text containing images into characters that can be readable by computers to edit, compute, and analyze in the future steps. Some popular use cases include digitizing books and other unstructured documents that enable human-human communication. One modern application is Google translate's OCR feature that enables users to scan anything and translate it, allowing them to practically read in any language.

Tesseract (https://tesseract-ocr.github.io) [8] is an OCR engine that was originally developed by Hewlett-Packard as proprietary software in the 1980s, which was released as open source in 2005 and development has been sponsored by Google since 2006.Tesseract supports APIs in several languages such as Python, Javascript, C++ etc.

An online OCR tool NewOCR [5] allows users to upload an image and extract the textual information contained therein. It supports multiple languages (up to 122) and different image formats There are several other proprietary OCR applications for Android and iOS such as ScannerPro [7] and Text Scanner [9].

Chrome extensions [3] are small software programs that customize the browsing experience. They let users tailor Chrome functionality and behavior in many ways, to improve productivity, enrich web page content, aggregate information and other applications.

PROPOSED INTERFACES

There are two interfaces proposed as part of this study, to extract textual information from images.

PicoloExtension

The first interface allows users to click a button to capture a portion of the screen and extracts the text from it. The extracted information is then copied to clipboard.

Design and flow of execution

The underlying design consists of a chrome extension that triggers the screen capture and allows a user to draw a bounding box. This is done in the background.js javascript code. Then the extension makes a POST API call with the byte representation of the extracted image, to a REST server that is hosting the OCR conversion. The REST server (hosted on Python using Flask framework) receives the POST request and extracts the image information from the payload. Then it processes the image to remove noise and make it suitable for OCR. Tesseract API is used to extract the text from the given processed image and then sent back to the client in the response payload. After the client (extension) receives the response, it copies the extracted text from the response payload to the clipboard. That terminates the flow.

PicoloDesk

This is a Python script which can be executed to capture the screen, draw a bounding box for the area of interest and then extract text from the image. It also saves the screenshot for the user.

Design and flow of execution

It consists of a Python script that captures the screen using "screencapture" and then uses OpenCV to allow the user to draw a bounding box for the region of interest. If there is a mistake, the user can redo the process by pressing the R key, otherwise proceed to OCR using the C key. This triggers the processing of the selected cropped image and uses Tesseract API to extract text from it.

EVALUATION

Heuristic Evaluation

As a first step, the design for PicoloExtension was evaluated based on the heuristics [6] and the feedback was incorporated into the final design. Evaluation feedback for the final design is shared next.

1. Visibility of system status: When a user clicks on the extension, the main screen fades and text "Capturing is Active" is displayed. Afterwards, when conversion is complete the screen gets restored and the text box goes away.
2. Match between system and the real world: The system screenshot design works in a similar manner and the user selects the boundary box.
3. User control and freedom: The user can abort the screenshot at any time and redo is easy to perform since this is an idempotent operation.
4. Consistency: The system behavior follows standard screenshot and clipboard operations.
5. Error prevention: The screen waits for a user to start selecting the bounding box and waits for them to finish selection.
6. Recognition over recall: This application does not require the user to enter any data.
7. Flexibility and efficiency: The user can see the bounding box as they draw it which acts as feedback.
8. Aesthetic and minimalist design: The interface fades the screen and displays a text box to show the user the capturing is active. The colors for the text and bounding box are chosen such that they would not be distracting or overwhelm the user.
9. Helps users recognise, diagnose and recover from errors: This interface does not involve any interaction where the user might make an error. E.g. If they make a mistake in drawing the bounding box and the text is incomplete, they can compare the output with expectation and redo the operation.
10. Help and documentation: The application provides detailed instructions for execution and efficient use.

Results

- The application can support a feature to display the extracted text on screen in addition to copying it to the clipboard. This has a severity rating of 1, i.e. a cosmetic issue.
- Recovery from errors: There can be an additional intelligence mechanism for the correction of spelling mistakes based on a language model's predictions. This has a severity rating of 2, which makes it a minor issue.

Usability Testing

A study was conducted with 4 subjects with within-subjects design due to limited availability of subjects. Each participant performed three tasks using each of the three interfaces. Their task completion time and error rates were measured. A summary is provided below and more details have been added to the appendix.

Tasks

- The first task was to copy/type a short command which consists of English language words without special characters.
- The second task was to copy/type a URL which would have special characters.
- The third task was to copy/type a short paragraph in English, without special characters.

Interface	Task 1		Task 2		Task 3	
	Mean completion time (sec)	Mean error rate	Mean completion time (sec)	Mean error rate	Mean completion time (sec)	Mean error rate
Manual	10.99	0	11.5675	0	27.47	1.375
PicoloExtension	8.495	0	7.2375	0	7.53	0
PicoloDesk	9.5775	0	9.6225	0	12.055	0.05

Figure 1. Mean completion time (in seconds) and error rate (percentage) for each of the interfaces across the three tasks.

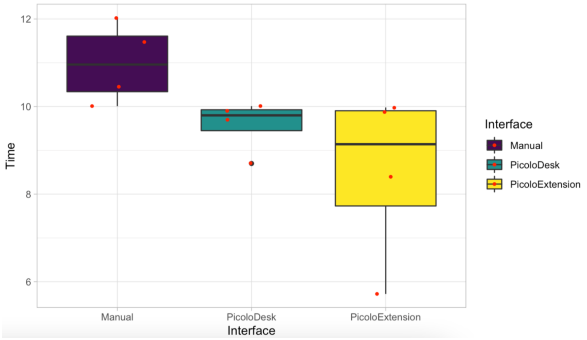

Figure 2. Descriptive statistics for the completion time for each of the interfaces for Task 1

Interfaces

- Manual input: all the subjects had to manually type the prompt they saw on the browser, in a text file open alongside the browser window.
- PicoloExtension: The subjects were supposed to use the chrome extension to select the area of interest and convert it to text. The result, which gets copied to clipboard, was then pasted to a text file.
- PicoloDesk: The subjects were supposed to execute the Picolo script which captures the screen and then asks the user to select the area of interest and then converts it to text. The result was then pasted to a text file.

Subjects

The participants performed these tasks in different order to counterbalance any learning effects from one task to another. They were familiarized with the hardware setup before conducting the experiment.

RESULTS

Descriptive statistics analysis

- For all the tasks, the mean completion time (seen in Figures 2, 3, 5 and summarised in Figure 1) is lowest for PicoloExtension interface and most for Manual interface. PicoloDesk interface's mean completion time is in between these two interfaces.
- There is a significant spread between the times taken for the Task 3 in the Manual interface as seen in Figure 5. This can be attributed to the fact that task 3 simulated a situation where subjects have to enter a small paragraph, so their typing speed is a factor.

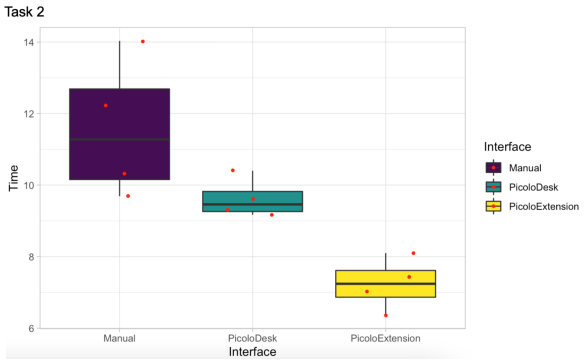

Figure 3. Descriptive statistics for the completion time for each of the interfaces for Task 2

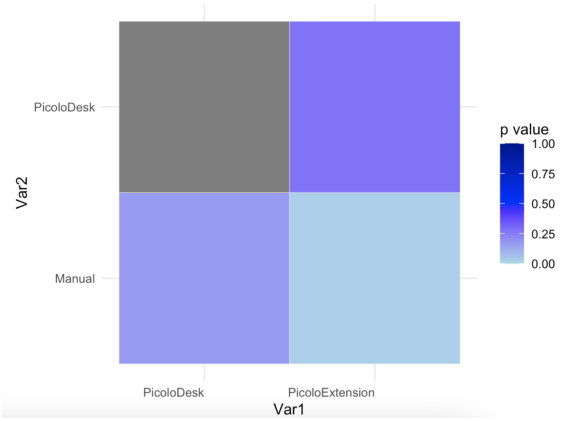

Figure 4. Pairwise t-test results for the completion time for Task 1

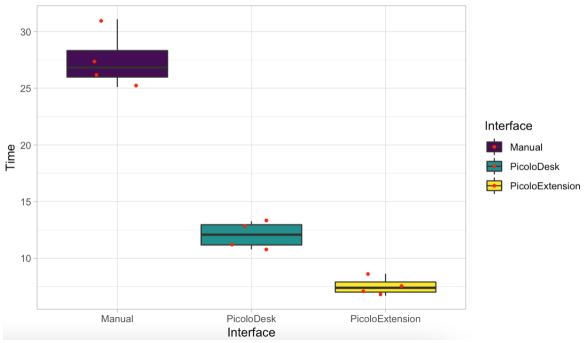

Figure 5. Descriptive statistics for the completion time for each of the interfaces for Task 3

Task	ANOVA test results for completion time
Task 1	Df Sum Sq Mean Sq F value Pr(>F) Interface 2 12.523 6.261 4.335 0.0684 . Residuals 6 8.667 1.445
Task 2	Df Sum Sq Mean Sq F value Pr(>F) Interface 2 37.63 18.813 14.44 0.00509 ** Residuals 6 7.82 1.303
Task 3	Df Sum Sq Mean Sq F value Pr(>F) Interface 2 874.3 437.1 173.7 4.89e-06 *** Residuals 6 15.1 2.5

Figure 6. ANOVA test results for the completion time for all tasks

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
Interface	2	4.865	2.4325	4.06	0.0767	.
Residuals	6	3.595	0.5992			

Figure 7. ANOVA test results for the error rate for Task 3

- The average time taken to complete Task 3 using different interfaces as seen in Figure 5 also shows that there is a bigger gap between the two Picolo designs and the manual interface. This highlights the efficiency of automated approaches when the task involves more volume of data to be extracted.
- The error rate statistics for Task 3 as shown in Figure 13 signify that humans not only tend to be slower, but also are more prone to making errors when manually entering a bigger chunk of data. This is where the PicoloExtension performs best.

Inferential statistics analysis

For each task, ANOVA (results shown in Figure 6 and Figure 7) and pairwise t-tests are performed. The independent variable is interface type, and the dependent variables are task completion time (in seconds) and error rate (percentage).

- For Task 1 completion time (as shown in Figure 2), the p-value for t-test between PicoloExtension and Manual interface is 0.024 which is less than the threshold value of 0.05, thus we can reject the null hypothesis, implying statistical significance between them. The same is observed in the heatmap shown in Figure 4.

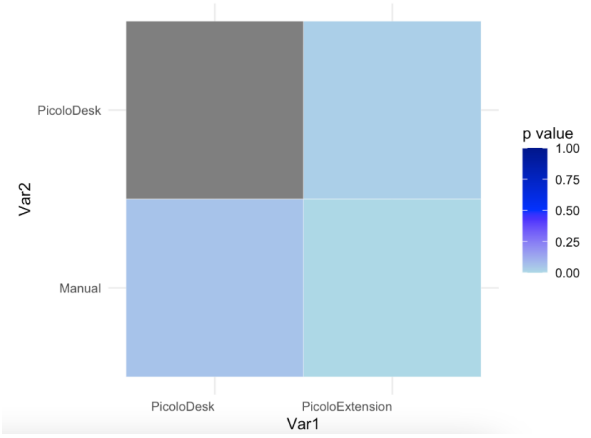

Figure 8. Pairwise t-test results for the completion time for Task 2

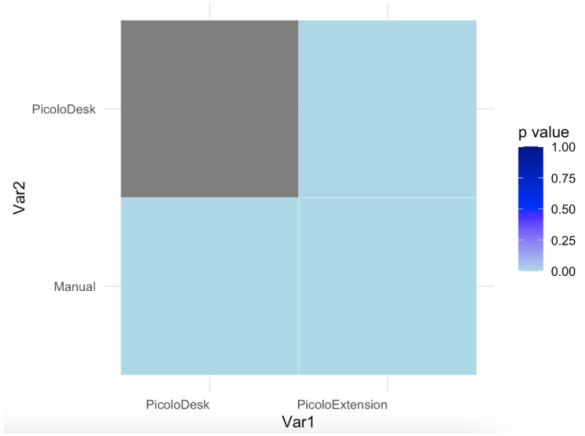

Figure 9. Pairwise t-test results for the completion time for Task 3

- For Task 2 completion time (as shown in Figure 3), the p-value for t-test between PicoloExtension and Manual interface is 0.00086 and for the pair PicoloExtension and PicoloDesk is 0.0247 (as shown in Figure 11) which is less than the threshold value of 0.05, thus we can reject the null hypothesis, implying statistical significance between these two pairs. The same is observed in the heatmap shown in Figure 8.
- For Task 3 completion time (as shown in Figure 5), the p-value for t-test between PicoloExtension and Manual interface is 5.073e-08, for the pair PicoloExtension and PicoloDesk is 0.0047, and for Manual and PicoloDesk it is 4.71e-07 (as shown in Figure 11), all of which are lesser than the threshold value of 0.05, thus we can reject the null hypothesis, implying statistical significance between all these pairs. The same is observed in the heatmap shown in Figure 9.
- For Task 3 error rates (as shown in Figure 13), the p-value for t-test between PicoloExtension and Manual interface is 0.037, and for Manual and PicoloDesk it is 0.043 (as shown in Figure 12), both of which are lesser than the threshold value of 0.05, thus we can reject the null hypothesis, implying statistical significance between these two pairs. The same is observed in the heatmap shown in Figure 14.

Overall we can see in Figure 10 that PicoloExtension interface outperforms the Manual interface on all the three tasks and there is statistical significance between these pairs based on the p-value and inferential statistics discussed above.

DISCUSSION AND FUTURE WORK

We observe that the proposed PicoloExtension and PicoloDesk interfaces outperform manual entry on all the tasks that were tested. However, one point to note is that it did not take into account a task with adversarial conditions. For instance, the behavior of this interface when the bounding box drawn by

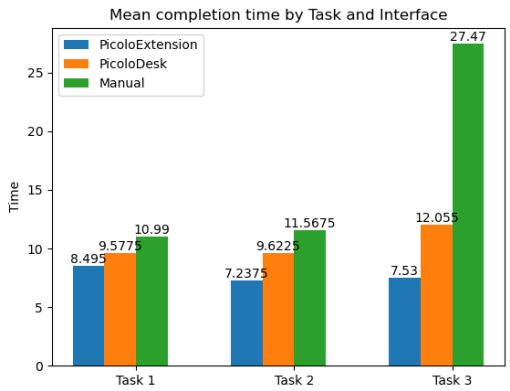

Figure 10. Comparison of the mean completion time for each of the interfaces across the three tasks.

Task	Pairwise t-test results for completion time
Task 1	Manual PicoloDesk PicoloDesk 0.16197394 NA PicoloExtension **0.02475508** 0.2729894
Task 2	Manual PicoloDesk PicoloDesk 0.0557494911 NA PicoloExtension **0.0008622376** **0.02470934**
Task 3	Manual PicoloDesk PicoloDesk **4.71128e-07** NA PicoloExtension **5.07292e-08** **0.004690057**

Figure 11. Pairwise t-test results (p-values) for the completion time for all tasks

	Manual	PicoloDesk
PicoloDesk	**0.04323557**	NA
PicoloExtension	**0.03738322**	0.9312502

Figure 12. Pairwise t-test results (p-values) for the error rate for Task 3

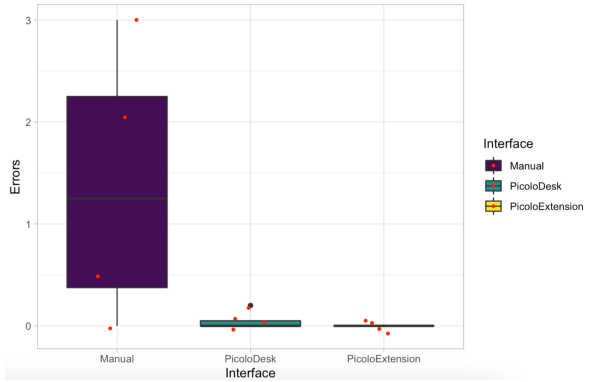

Figure 13. Descriptive statistics for the error rate for each of the interfaces for Task 3

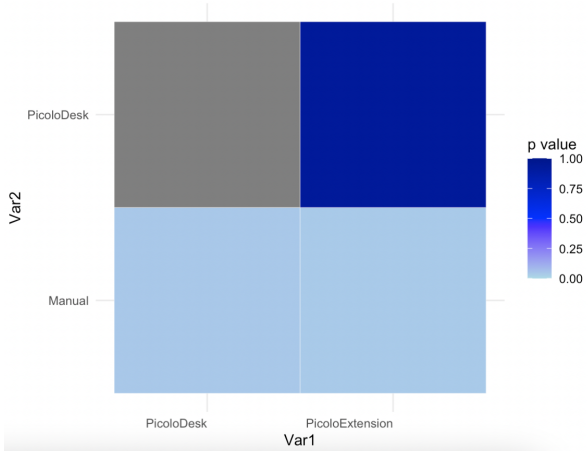

Figure 14. Pairwise t-test results for the error rate for Task 3

the user is not accurate, such as splitting the text horizontally. In such cases it was observed that the proposed techniques performed significantly worse than humans, who are able to somewhat correctly identify the text. Another aspect that has a similar issue is the resolution of image captured, as OCR is known to be sensitive to the image quality and noise.

This motivates that a component of intelligence be added to the OCR server by leveraging language models such as the recently proposed pretrained transformer-based language model GPT3 [2]. This would afford error tolerant extraction of text in adversarial conditions such as the ones discussed in the previous section.

As an extension, support for extraction of text from a video stream can be useful for various applications such as scribing notes for a lecture, generating the minutes of a meeting etc. Future integration by creating plugins compatible with latest video conferencing applications can enhance the user experience and productivity. Furthermore, Tesseract supports OCR in multiple languages which can be integrated in this tool, allowing users to identify text in more contexts.

CONCLUSION

To improve the user interaction experience in online classes and meetings, a smart assistant: Picolo, is introduced. Two design interfaces are provided - PicoloExtension and PicoloDesk that leverage OCR (Optical Character Recognition) to help us extract textual information from images, in real time. We provide a comprehensive evaluation of these interfaces against the manual entry interface (and each other), using discount usability testing and user testing. We analyze the results of descriptive and inferential statistics on the evaluation study, proving significant performance gains using the proposed PicoloExtension. Additionally, we discuss enhancements and propose directions for future work.

ACKNOWLEDGMENT

The author would like to thank the instructor and teaching assistants of the course CSE518 Human Computer Interaction

Figure 15. The three tasks performed by users as part of the usability testing experiment.

for their invaluable teachings, guidance and feedback that motivated and helped in designing this project.

REFERENCES

[1] Tanvi Aggarwal. 2021. Source code for Picolo: A Smart Assistant. (2021). https://github.com/TanviAgg/PicoloExtension.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1877–1901. https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[3] Chrome extensions 2021. Google chrome extensions. (2021). https://developer.chrome.com/docs/extensions/mv3/overview/.

[4] Golang Text source code 2021. Golang Text processing support package (readme used for experiment). (2021). https://github.com/golang/text#readme.

[5] New OCR 2021. Online OCR Tool: New OCR. (2021). https://www.newocr.com.

[6] Jakob Nielsen. 1994. Enhancing the Explanatory Power of Usability Heuristics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '94)*. Association for Computing Machinery, New York, NY, USA, 152–158. DOI: http://dx.doi.org/10.1145/191666.191729

[7] Scanner Pro 2021. Scanner Pro OCR for iOS. (2021). https://apps.apple.com/us/app/scanner-pro-pdf-scanner-app/.

[8] R. Smith. 2007. An Overview of the Tesseract OCR Engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, Vol. 2. 629–633. DOI: http://dx.doi.org/10.1109/ICDAR.2007.4376991

[9] Text Scanner 2021. Text Scanner OCR for Android. (2021). https://play.google.com/store/apps/details?id=image.to.text.ocr.

APPENDIX

Resources

Source code for the chrome extension and desktop application is available at https://github.com/TanviAgg/PicoloExtension. Technical references are cited in the readme.

Usability testing experiment design

The experiment defined three tasks to be performed by each user using all the interfaces, as shown in Figure 15. The data was extracted from the following webpage during this experiment: https://github.com/golang/text#readme [4].

Instructions for PicoloExtension

Setup

1. Download the source code from the repository [1].
2. Chrome extension:
   - Go to Chrome > Manage Extensions > Load unpacked.
   - Upload the PicoloExtension directory which contains all the required files for the chrome extension.
3. OCR server - Start the REST server using the following commands:

```
export FLASK_ENV=development
export FLASK_APP=basic_server.py
flask run
```

4. Now the server is ready (by default started on port 5000), and the chrome extension can be used to extract text from images.

How to use

1. Pin the extension to the toolbar and click on it. The screen will display a message saying "Capturing is Active".
2. Select the bounding box for the area of interest using the cursor.
3. After the cursor is released, the converted text will be copied to the clipboard, for use as needed.

Instructions for PicoloDesk

1. Follow the instructions given in the readme for environment setup.
2. Execute the script main.py and it will display a pop-up with the captured screenshot.
3. Using the cursor select the part which you want to extract text from, and a green bounding box will appear.
4. If you want to redo the selection, click the R key and go back to step 3.
5. Otherwise you can click on the C key, which will select the area and extract the text from it. Press any key to exit.