Pipeline construction for the automated text retrieval, editing, and deletion in comic illustrations

Jordi van Setten[1]

Supervisor(s): Lydia Chen, Zilong Zhao

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Jordi van Setten
Final project course: CSE3000 Research Project
Thesis committee: Lydia Chen, Zilong Zhao, Anna Lukina

Abstract

With the increasing demand for high-quality data in the field of Machine Learning and AI, the availability of such data has become a major bottleneck for further advancements. This paper proposes a novel approach to extract valuable data from comic illustrations, aiming to address the scarcity of labeled datasets. By leveraging popular comic series such as Dilbert, which contain thousands of comic strips with multiple panels, text boxes, characters, and settings, we aim to create a pipeline for data labeling and manipulation. This pipeline will enable experiments in various areas, including generative comics, humor detection, translation, and more.

The paper focuses on two key research questions: 1) How accurately can we get current OCR models to extract text from the comics, and 2) How can we create the ability to edit and delete existing text boxes. By accurately segmenting the panels and text boxes within the comics, we expect to improve OCR performance by reducing noise and addressing unique text formats. Object detection models will be employed to zoom into text boxes, further enhancing OCR text extraction accuracy. Evaluation metrics such as Latent Dirichlet allocation (LDA), Character Error Rate (CER), and Word Error Rate (WER) will be used to measure the effectiveness of the proposed techniques.

In the end, utilizing a dataset of 500 labelled comic panels, we achieve accuracies of 94.07% for CER (up 9.86% from 84.21% baseline), 88.35% for WER (up 8.35% from 80.0% baseline), and 98.0% for LDA (up 3.77% from 94.23% baseline). Similarly, editing and deleting of text boxes inside the comic panels prove to be successful in a vast majority of instances. We believe these results are more than adequate for select use cases.

1 Introduction

With the rapid acceleration of Machine Learning and AI in today's world, the demand for high quality data is increasing significantly [1][7]. Machine Learning models are often dictated by the quality of the data that they are provided with, and as such the quote of "garbage in garbage out" has been coined. [20][18]. Currently, there is a severe lack of high quality data to train on, often due to the need to manually label large parts of the data. This results in major bottlenecks for the further improvement of models [10].

The effectiveness of data construction pipelines has been shown previously in works such as "A data-driven approach to cleaning large face datasets"[11]. Utilizing the created data allowed for the further advancement of face recognition research.

In this paper we look towards a novel area of comic illustrations to extract valuable sources of data. Comic series such as Dilbert, PHD Comics, and Garfield, contain thousands of comic strips, each with multiple panels, text boxes, characters, and settings. Extracting the high quality data will allow for experiments to be conducted in areas such as facial recognition, generative comics, humor detection, translation, and more.

- How accurately can we get current OCR models to extract text from the comics
- How can we create the ability to edit and delete existing text boxes

The end result will be the creation of a pipeline that is able to label comics with the text associated to each panel, and also allow for the editing and deletion of the text in the comic. The end result will give access to much larger data sets and allow for rapid improvements in computer vision.

- Segment the panels of the comic
- Segment the text boxes from the panels
- Extract the text from the text boxes

The reason why we believe that this method improves the overall pipeline is due to the significant reduction in noise. Noise in the form of background colors, unique text format, etc, is the biggest hindrance towards OCR accuracy[3]. Segmenting the panels is an effective step, used by Maciej Styczen, which eliminates the panel barriers. We propose taking this even further by zooming into the text boxes using object detection models. We hypothesize that this will improve the results even further during the OCR text extraction phase of the pipeline.

We will measure the accuracy of this technique using Latent Dirichlet allocation (LDA), Character Error Rate (CER), and Word Error Rate (WER). LDA aims to measure the semantic meaning behind a sentence and compare how much the two relate to each other. CER and WER both aim to measure how many insertions, deletions, and substitutions of characters/words are minimally necessary in order to achieve the ground truth result.

This research builds upon the prior work done by Maciej Styczen who has already been able to successfully build a pipeline to extract the data. Our goal is to significantly improve the accuracy of the

Figure 1:
Old method: "isaidi you mean you guessed?"
New method: "I said I winged it. You mean you guessed?"

Figure 2:
Old method: "thats because you are already a genius that but dont sounds know wir right."
New method: "thats because you are already a genius but dont know it. that sounds right."

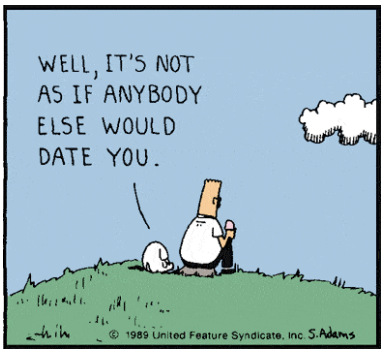


Figure 3:
Old method: "well, its not as if anybody else would date you. v tin den this i .. united feature syndicate, inc. s. adams"
New method: "well, its not as if anybody else would date you."

output. Some examples where the old method fails and the new one succeeds are shown in figures 1 2 3.

Each example highlights an issue the old method faced that the new method resolves. In figure 1 we can see the old method incorrectly concatenates "I said I" into "isaidi", on top of also completely missing "winged it." Figure 2 shows an issue where it cannot differentiate the two different text boxes and reads it as one, hence the ending result of "a genius that but dont sounds know wir right." where a mistake is also made with "it", being seen as "wir". Lastly, the old method in figure 3 correctly detects the text, however also detects text where there is none or is unintended, such as the publication and author at the bottom, but also "text" in the grass (which the model believes is "v tin den this i". Our new proposed method will be able to fix all of these mistakes and produce a much stronger output.

2 Related Work

For the related work we look towards the many topics we are tackling and evaluate them thoroughly in order to achieve a successful pipeline. The topics are all the ones related to the pipeline: panel segmentation, text boxes location, and text extraction.

Panel Segmentation. When tackling panel segmentation, the subjective nature of comics and the artistic design choices of the artists creates a difficult challenge to achieve accurate results. Despite this, there are studies that have tackled this exact problem with various techniques such as Convolutional Neural Networks (CNN) [12], and contour detection algorithms [5][14]. CNN's have promising results, with estimates of 97% accuracy [12]. However, the issue it faces is that it requires a large amount of labelled data in order to fine tune effectively. When we look at the research done by Maciej Styczen, his method of utilizing the contour detection algorithm together with binarization and adaptive thresholding proved to be extremely effective with a 100% accuracy, when tested on 1118 panels. This is even better than the 97% achieved by the "Kumiko, the Comics Cutter" repository [8] which also uses the contour detection algorithm.

Text Segmentation. Segmenting text boxes is something that has been researched to some extent before, however not thoroughly. The most promising method appears to be the use of object detection models. One paper combined this with OCR, with the goal of identifying Pakistan vehicle number plates [19]. However, in our use case, problems may arise due to the varying nature of text boxes. Some appear within little "clouds", clearly indicating they are text boxes (An example could be the comic in Figure 1), and some which have them blended into the background (e.g. Figure 2). To address this, fine tuning has been proven to be an effective method of capturing objects which most standard object detection models are not trained on. [13]. The potential downside of this strategy is that it may not generalize to other comics well due to the various art styles different comics have. Another research paper proposes to take this even further by adding a novel scene text detection method that combines the localization of corner points of text bounding boxes with the segmentation of text regions [9]. The ultimate purpose of this is to allow for text that is curved or slanted to be interpreted correctly.

Text Extraction. Current models work very accurately when taking in some standardized text. However, dealing with colorful backgrounds, handwritten texts, and small fonts is something that proves to be difficult [3]. To overcome this, current studies suggest fine tuning the OCR models to the specific context you want to apply them to [17]. Furthermore, using error correction and text validation [16] is an effective strategy, since minor mistakes such as "i" and "l" can be misinterpreted by OCR. However, with the words "laugh" and "iaugh" it becomes apparent that "laugh" was intended. Lastly, preprocessing techniques such as upscaling and binarizing the image are also effective strategies [2]. This is due to the optimal character height for OCR being 20-40 pixels[21], which in a comic is not always the case. Moreover, binarization removes a lot of excessive noise created by background colors, which is known to be a hindrance for OCR [3].

This research is built upon the work of Maciej Styczen [21] with the research titled "Automated Text-Image Comic Dataset Construction". There are a number of techniques that are directly taken from the prior work done. These include the panel segmentation, pre-processing of images, and OCR models used. The current state of the research leaves a lot of inaccuracies on the table, and also does not provide an effective way to edit and delete text boxes. As such, in this research project we are expanding upon this work by adopting the use of object detection models for the purpose of improving OCR accuracy and adding the ability to edit and delete text.

3 Methodology

In this section we will explain how we create our pipeline that can take in an arbitrary amount of Dilbert comic strips and return a CSV output containing the panels inside the comic, and the associated text. We will look into panel segmentation, text extraction, OCR models, evaluation metrics, and editing/deletion of text boxes.

3.1 Extracting Text from Comics

Our research question aims to find the most accurate method of retrieving text from a comic strip. To achieve this, we will experiment with various techniques and combinations. We will compare these combinations to determine the most accurate result.

For panel segmentation, we will use the contour detection algorithm as used by Maciej Styczen, as it is reliable and generalizable to various comics as seen in the abstract of Figure 8 [21]. Effectively, it is a process that involves first binarizing the image, as seen in Figure 8b. Using this we can find the outermost contours, as shown in Figure 8d. The next step is to have the inside of the region marked by the contours, colored white. Finally, the remaining text is removed by utilizing morphological opening operations. This combines both dilation and erosion in order to remove the unwanted text, and preserve the shape.

When considering text box detection, we will use an object detection model as this is supported by numerous studies. We can then consider using an out of the box, fine tuned, or utilize localization of corner points. In our use case, the last one will have no benefit, and could only serve to reduce accuracy. This is because its intended purpose is to concatenate non-linear text together, however the vast majority of text we are dealing with is linear.

Choosing between fine tuning or not is a difficult choice to make, since it is hard to judge whether the increase in accuracy with finetuning will outweigh the potential reduction in generalizability. As such, both will be tested and compared in order to scientifically evaluate the results and draw conclusions based upon it. Vision API provides an out of the box "ImageAnnotatorClient" which we will use. For the finetuning, approximately 500 comic panels (with 1 comic strip having on average 3-4 panels) are taken and manually labelled using Google Cloud's Vertex AI. Using this same platform we are able to fine tune its existing object detection model with our data. After approximately 2 hours of training, and deploying the model, we will be able to utilize the fine tuned model. It will take as input a comic panel, and output the resulting bounding boxes (can be multiple).

OCR With the text boxes located, the next step is to take the text boxes and conduct OCR on them directly. For this we will test the effectiveness of both Google Tesseract and Google Vision API and compare the results. These two are arguably the most popular models at the moment, with Tesseract being completely open source, and Vision API being the state-of-the-art commercial model. We expect Vision API to perform better, however there are also costs associated with this, through general pricing for the usage, but also run time. This is due to Tesseract being installed locally on your device, whereas Vision API is used through API calls, which is inherently slower and more expensive.

Finally, we will need to stitch the text back together in the correct order. This can be quite the challenge due to comics sometimes having an ambiguous ordering of text boxes (see Figure 4). However, we can assume that the text will most reasonably read from top left to bottom right. As such, we can use heuristics to help us find the most likely correct order. This starts by looking at the top-most text and selecting that as the first text box. If there are multiple text boxes at the top, we will prioritize the leftmost

one. However, due to text-boxes being imperfectly placed and located, this could result in a top left text box (which is intended to be read first), be surpassed by another text box at the top right, due to it being placed just a few pixels above (see Figure 5). To combat this, we will create a "window" within which the highest "y" coordinate must fall. In other words, we will find the highest "y" coordinate, then look 10-100 pixels below it, and put any other text boxes found into the same "bucket". These will then be sorted on the "x" coordinate, with the smallest x value (left most box in image) taking the priority.

Evalution metrics Having all this in place, there will be six combinations we will evaluate. They will be evaluated in order to establish the effectiveness of utilizing object detection for textboxes, finetuning this object detection model, and also evaluating the impact that using Tesseract has over Vision API.

To measure this accuracy we have 500 unseen labelled data to test our results on. We will be using various accuracy metrics such as Character Error Rate, Word Error Rate, and Latent Dirichlet allocation. Character Error Rate (or Levenshtein distance) and Word Error Rate measures the minimum number of substitutions, deletions, and insertions of characters/words necessary for two strings to become identical, and divides that by the total number of characters/words in the ground truth. For consistency of results, we subtract the result from 100 in order to make it clear that the higher the result, the better the outcome.

Latent Dirichlet allocation takes a different approach. It aims to infer the topic distributions for each text provided and compare the two. This process is performed using probabilistic inference techniques, such as variational inference or Markov chain Monte Carlo sampling.

When considering the implication of the accuracy metrics, the first two techniques highlight the overall raw accuracy of the extraction. There are many scenarios where you would want this metric to be high, for example training another OCR model. Having incorrect output in this scenario, would cause a new OCR model to be incorrectly trained. For the LDA metric, the semantic meaning behind the text is measured. Take for example an application that aims to translate comics into another language. Having a very high score on LDA would imply that most, if not all, of the semantic meaning will be captured, allowing for effective translation to take place. These are just a few examples, and the importance placed on each metric will have to be evaluated per individual use case.

3.2 Replacing existing text

The aim is to build the functionality to edit and remove text from comics. In order to achieve this we will need to first detect the location of the text boxes, and then insert a new text box on top of it with the desired text. Finding the location of the text boxes will utilize the same object detection models as in the previous part, since they are designed to do exactly that. Depending on the results of the previous part we will choose whether or not to use a fine tuned model for this.

In order to determine which color the new text box will be, opencv will be used to count the number of occurrences of every color in every pixel inside the bounding box. Then using opencv again, we insert this box over the original image, resulting in us effectively "deleting" the previous text as seen in Figure 6

In order to "edit" new text, instead of inserting a new box directly over the bounding box, we first insert the new text into the new box. This is not entirely trivial, due to varying lengths of text, and varying shapes and sizes of text boxes. As such, we need to correctly wrap the text to fit any arbitrarily sized text box provided. To do this we make use of the text wrap library, which allows us to input our desired text and the width of the text box, and give us an output of a list of strings. We then iterate over this list and insert every string line by line using $opencv.putText()$, with the "y" position being incremented by a variable $line_height$. Once this is complete, we have our new text box and all that is left is to insert it over the original bounding box. The user will be able to input whatever text they would like, and as such, give us the ability to "edit" comic images as seen in Figure 9.

The proposed methods do come with some limitations. The biggest one being that text boxes are not always fully rectangular, whereas the object detection model's bounding boxes are. This will often result in areas getting overwritten that are not fully intended as can later be seen in Figure 10. Moreover, text boxes with backgrounds being in different colors (as is frequently seen in the newer comics), will result in odd looking outputs as seen in Figure 11.

4 Evaluation

In order to evaluate our results, we will be using 500 unseen, but labeled, Dilbert comic panels. These 500 panels will be run through the created pipeline with various parameters, and the output text will be compared with the ground truth. These two resulting text strings will first be cleaned from all non-alphanumeric values (such as "," "." "?" "!" etc.) This is due to them having a minor impact in determining semantic meaning behind the sentence, but attribute to a significant amount of mistakes. The parameters that we will test include the OCR Model (Tesseract or Vision API), and the Object Detection model (None,

Figure 4: Ambiguous text box ordering

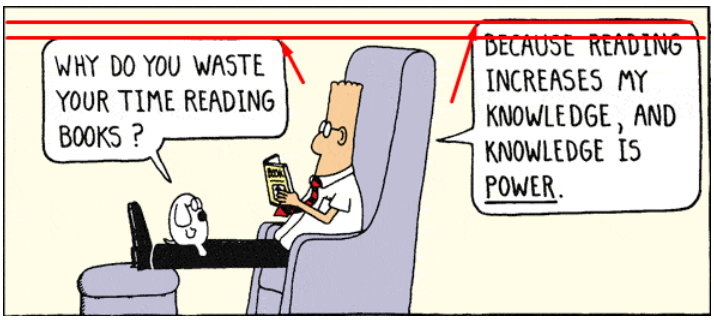


Figure 5: Incorrect ordering based on height

General, or Fine tuned). As such, our combinations are:

- No Object Detection + Tesseract
- No Object Detection + Vision-api
- Object Detection + Tesseract
- Object Detection + Vision-api
- Fine tuned Object Detection + Tesseract
- Fine tuned Object Detection + Vision-api

Each combination will be evaluated on all three metrics, Character Error Rate (CER), Word Error Rate (WER), and Latent Dirichlet allocation (LDA).

4.1 Results

Text retrieval results

We have run a total of 6 experiments with every single combination possible, as seen in Table 1. Table 2 highlights which experiment yielded the best results.

Looking at LDA, we can see that the first three experiments, which used Tesseract, all performed worse than the latter three, which used Vision API. On top of this, using a fine tuned model proved to give the most accurate results in all instances. A fine tuned model together with Tesseract gave an accuracy of 92.8%, and with Vision API gave 98.0%. Meanwhile, not using any object detection gave the worst results for their respective OCR model, with Tesseract and Vision API getting 89.06% and 94.23% respectively. When it comes to LDA it becomes apparent that the proposed fine tuned object detection model hits the mark, and is a clear favorite.

Looking towards CER we can see very similar outcomes. Using a fine tuned object detection model always performed better than using a general one or none at all. In this case we can even observe that when the fine tuned model is combined with Tesseract, it will outperform Vision API when it is not using any object detection model (86.38% vs 84.21% respectively). We can also observe very similar trends when looking at WER

Overall, we can clearly observe that in all areas, using a fine-tuned model together with Vision API yields the best results.

Editing and Deletion results

We can see some examples of outputs in figures 6, 9, 10, 11. Figure 6, shows the process of removing the text with the use of text box location and then text insertion into the text box. Figure 9 shows another example where there are no text boxes present, just "floating" text in order to display the successful use in this situation as well. However, figure 10 shows complications which arise when the text box does not have a rectangular shape. Finally figure 11 displays the issues with the modern comic panels which have a sort background "fade" to them and how it makes the next text box stand out.

To evaluate the accuracy of these results, the most practical thing to do is look at the accuracy of the object detection model. This is because for our editing we are inserting and deleting boxes directly over the indicated text box location, which is retrieved by the object detection model. With a labeled data set of 360 comic panels, there are approximately 560 text boxes. Having a 10% validation set gives us 56 comic panels to test our accuracy on.

Given this information, we have a recall precision of 92.5%, meaning that in 92.5% of the cases, the model correctly identified the presence of a text box in a comic panel. Interestingly, all the errors came from instances where the text was much different looking than normal. Take for example the comic panel in Figure 12, eventhough the model managed to capture "THE EXERCIST!", it missed the "AAAGH!", which is much larger than the expected text and is also slanted. Another example is in Figure 13, where both the "UH OH" and the "DING DONG" were not identified correctly. The "UH OH" is clearly much smaller in size, which may

Table 1: Model and Object Detection Comparison

Exp. no.	OCR Model		Object Detection Model		
	Tesseract	Vision API	None	General	Fine tuned
Exp. #1	✓		✓		
Exp. #2	✓			✓	
Exp. #3	✓				✓
Exp. #4		✓	✓		
Exp. #5		✓		✓	
Exp. #6		✓			✓

Table 2: Performance Metrics Comparison

Exp. no.	CER	WER	LDA
Exp. #1	81.97%	69.26%	89.06%
Exp. #2	84.2%	64.11%	90.29%
Exp. #3	86.38%	71.02%	92.8%
Exp. #4	84.21%	80.0%	94.23%
Exp. #5	87.51%	81.47%	95.77%
Exp. #6	**94.07%**	**88.35%**	**98.0%**

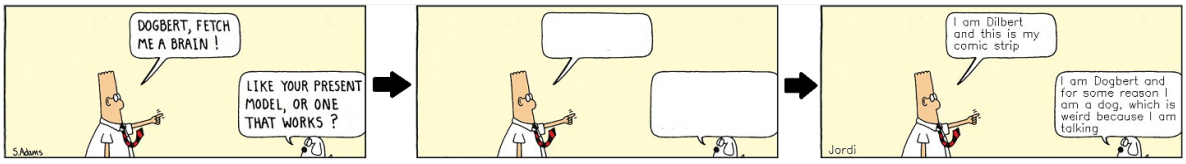


Figure 6: Editing and Deleting a comic panel

be the reason why it was missed. The "DING DONG" likely suffers the same problem as "AAAGH!", with it being slanted and not entirely resembling a text box.

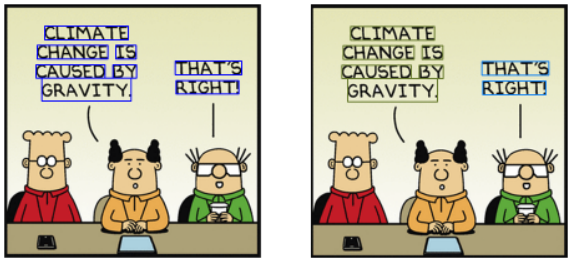
We can also further analyze the confidence associated with various results as seen in Figure 14. As we can see, the lowest score is associated with the "WUMP" with a score of 0.718. It follows similar issues as described above, with it being slanted and not being in the common text box type. In the same Figure we can see other slightly odd text boxes, such as the yellow bar at the top in the second panel which has a confidence of 0.889. It is again not directly a text box, but just text giving some additional information, and could be the reason for its poorer confidence. The next few lowest ones are from the panels which have relatively small text boxes. This highlights again that the smaller text boxes are much more difficult to identify, with a confidence of 0.925.

Lastly, looking at the other end of that spectrum in Figure 15, we can observe that all of these have a confidence of 0.994 or higher, and they are all clear cut text boxes as you would expect. They are also large, and don't have any odd backgrounds to obscure any details.

4.2 Discussion

Text retrieval

From the observed results we can clearly see that fine tuned text box detection together with Vision API have the strongest accuracy in every category of measurement on LDA. Analyzing the results, we can see some indications as to why this may be the case. When not using any object detection model or a non-fine tuned one, the OCR model will frequently find unintended text areas, for example author signatures, page numbers, and even grass, as seen in Figure 3.



(a) All b-boxes (b) Identified clusters

Figure 7: Example of bounding-box clustering [21]

This will have a huge impact on the accuracy of all metrics, due to there being extra words and characters that will directly impact WER and CER results, but almost certainly also impact the overall semantics of the sentence, and as such impact LDA results. The fine tuned model aims to tackle exactly this problem, and has clearly proven to be effective as it improves every accuracy metric in all instances.

Another observation is that when not using an object detection model, the ordering of the text is much more frequently observed to be in the incorrect order, as seen in Figure 2. When not using text boxes, clustering is used to identify which text boxes belong together as seen in Figure 7 [21]. Although effective in its use case, this does not solve the problem of one text box being slightly below another, despite being intended to be read first, as we observe in Figure 5. We solve this problem by identifying the text boxes and applying our heuristics text box ordering algorithm, as described in the methodology, in order to maximize this accuracy.

When looking at Tesseract vs Vision API it is clear that Vision API performs better. Even when

comparing the seemingly best option of fine tuned text box, using this together with Tesseract will have it perform worse than using no text box with Vision API in almost every accuracy metric.

However, when using Vision API, we do need to consider the drawbacks of pricing and run time. First, running Vision API costs $1.50 per 1,000 images [4]. This may seem relatively cheap, however, considering that Dilbert alone has approximately 12,384 comics, with every comic having roughly 3 panels, equating to approximately 37,000 panels, of which again there exist nearly 2 text boxes per panel, bringing the total images to be read up to 74,000. This will ultimately cost $111.00 to run a single time on all the data and should be factored in in any decision making.

Moreover, when comparing running times, we observed Tesseract having an approximate run time of 0.3 seconds per image, whereas Vision API took approximately 3 seconds per image, a 10x slowdown. This is due to Tesseract running locally on a machine, whereas Vision API needs to go through the network. However, there are factors to consider, such as the machine you are running it on, the internet connection speeds, etc. which may affect these speeds. There are also options to parallelize both Tesseract and the API calls, together with doing some batch processing, which may also impact speeds.

Ultimately, depending on your needs, if the cost is not an issue, and accuracy is of extreme importance (as we suspect it will be), further exploring parallelization techniques, and employing batch processing should significantly reduce the time taken per image for both OCR models. If you are limited by costs and speed, with accuracy not being as critical, using a powerful machine with Tesseract would be the most viable option. Otherwise Vision API is the clear favorite.

Image text editing and deletion

To look at the successes and failures of this section, we can look at some figures. In Figure 6 we can see the process of removing the text from the text boxes, and inserting new text into the text box. The text boxes are neatly preserved in order for new text to be flawlessly added in and the text wraps in accordance to the width of the text box. If we look at figure 9 we can see that even when there are no text bubbles "clouds" and varying text widths, the image can still be well preserved and have new text inserted.

However, when we look at figure 10 we can see some issues begin to arise. Although relatively rare, there are comics where the text boxes have a non-rectangular shape. Since the object detection model is purposed towards rectangles it will result in unintended areas to be overwritten and as such, create basically unusable panels.

Another issue, which unfortunately appears more frequently, is seen in Figure 11. The more modern comics have a sort of "fade" going on in the background. Again, working with the opencv library we are only able to insert full rectangles into these text boxes and as such, get odd looking rectangles that clearly stand out. Luckily, the majority of comic panels do not use this "fade", with the first year this style appearing being in 2008. This means that there is still almost 20 years worth of comics (starting from 1989) that will work as intended.

There are two ideas to overcome this that will also be discussed further in the future work section. The first idea is to utilize Maciej Styczen contour detection algorithm, specifically the part where mathematical morphology is used to scrub away the text, but now specifically inside a text box. The second is to use generative AI editing, by removing the detected text boxes and letting the AI replace them with empty text boxes that will fit the surrounding background and shape.

Lastly, looking at the results of the object detection model in Figures 12, 13, 14, and 15, it is clear that the issues arise with "unique" looking text, such as seen in Figure 12 with "AAAGH!" being larger and slanted, and smaller text as seen in Figure 13 and some of Figure 14. All these issues should be resolvable by further fine tuning the model, and providing it with more labelled data that has instances similar to the one it fails or performs poorly on. Doing this would improve the editing/deleting functionality, but also further improve the text retrieval and as such is an important step to take in the future.

5 Conclusion

In conclusion, throughout our research we had the goal of answering our two research questions:

- How accurately can we get current OCR models to extract text from the comics
- How can we create the ability to edit and delete existing text boxes

In the end, we believe we managed to answer them both effectively, while simultaneously providing context to certain limitations of the research.

5.1 How accurately can we get current OCR models to extract text from the comics

For this research question we will first acknowledge that we are able to successfully extract the text from numerous Dilbert comic strips. In terms of accuracy, our answer would be that with using Vision API, together with a fine tuned object detection model for text box location, we achieve significantly better results than the base case of Vision API and no text

boxes. For comparison, we achieved an accuracy of 94.07% for CER (up 9.86% from 84.21%), 88.35% for WER (up 8.35% from 80.0%), and 98.0% for LDA (up 3.77% from 94.23%).

With LDA being significantly high, we can confidently recommend using the outputs of these results towards tasks that require accurate semantic meaning of the comics, for example translation tasks. For CER, with an accuracy of 94.07%, according to benchmarks set by Holley, Rose [6], we would be considered "average", and as such should be used in practice with caution. We would still encourage further development and research in order to achieve higher accuracy results before using it on a large scale. Similarly for WER, we see a massive improvement from the baseline, but still advise against the usage on a large scale.

5.2 How can we create the ability to edit and delete existing text boxes

For this research question we can evaluate the outputs seen in several figures such as 6, 9, 10, and 11. We can also evaluate the accuracy of the object detection model as it is the primary indication of whether the text boxes will be accurately overwritten or not.

The object detection model achieves an accuracy of 92.5% when evaluated on the validation set of 56. The mistakes primarily arise under circumstances where the text is either very small in terms of font size (Figure 13), small in terms of text box size (Figure 13), or being a uniquely formatted piece of text (Figure 12). Although 92.5% is definitely high, with further fine tuning on these specific edge cases, we believe we can get very close to 100% accuracy.

In terms of actual output, there are issues that arise when the text box is not rectangular, as seen in Figure 10, or when the background has a "fade" to it, as seen in Figure 11. These issues cannot currently be overcome, but there are some promising ideas that would allow for these to be processed effectively as well. Nonetheless, with the current implementation, with all the Dilbert comics before the year 2008, the "fade" issue does not exist. As for the non-rectangular issue, these occur very rarely, and as such the functionality will still be useful in many use cases.

With all this in mind, we believe that we have been able to successfully add in the functionality of editing and deleting text from existing comic strips, and make it usable for any intended purpose, one such being providing data for fine tuning generative AI on Dilbert comics.

5.3 Future work

Text Retrieval

To further improve upon text retrieval, there are several ideas to explore. Despite the effectiveness of Vision API, sometimes it still misreads certain characters. As such, fine tuning an OCR model could prove to completely eliminate any errors that may occur as a result of this.

Further, fine tuning the object detection model even deeper, in order to capture every single intended text box, including the smaller sized, smaller fonted, and "unique" looking ones, would push the accuracy of the text retrieval even further.

We believe that improving upon all these elements should see a near perfect text retrieval process occur.

Image text editing and deletion

There are several ideas to consider here as mentioned before, such as further fine tuning the object detection model, using generative AI to edit, and using mathematical morphology on text boxes.

Further fine tuning the object detection model will serve to improve both text retrieval and image text editing/deletion. It is essential in the process of locating where the text boxes are in order to remove them or edit them.

Once we find the text boxes, one idea is to use generative AI editing in order to overcome the shortcomings of non-rectangular text boxes. Using DALL-E 2, we can see some preliminary results of what this could look like. Take for example the issue in Figure 10, using DALL-E 2, we can create something that looks like what we see in Figure 16. There are of course many issues, such as Dilbert not quite looking like Dilbert, and some variations have text filled in, however, it does manage to correctly create the empty text box in other instances, and preserve the background with the door and wall color. Another example is from Figure 11, which has the issue of a "fade" in the background. Running DALL-E 2 on this yields the results seen in Figure 17. Again, we can see some very strange images with Dilbert having 2 heads, strange artifacts that don't really make sense, but we can importantly observe that the background preserves the "fade" which we originally lost. With some significant work, this could be an effective way to edit and delete text boxes whilst preserving almost every detail.

Lastly, In Maciej Styczen work on the contour detection algorithm for panel segmentation, mathematical morphology was used in order to remove some unwanted text in the panels. We believe the same philosophy could be applied to the case of removing text from a text box whilst preserving backgrounds. There are some concerns however that it will also destroy some of the backgrounds through the "erosion" process. However, it is hard to predict how much the erosion process will affect it, and thus needs to be studied further.

6 Responsible Research

6.1 Reproducibility

To ensure the reproducibility of this research, all the code will be pushed onto Github, and extensive instructions left in the readme. Moreover, due to the large volume of the data, it cannot be pushed onto Github, but will be available upon request, in order to ensure that everything can be reproduced by other users. The code will also be forkable to allow future users to continue developing on the research.

6.2 Integrity

To ensure the integrity of the results, every test was run multiple times, with careful inspection and upon the results to ensure their validity. The labeled data set has also been verified for its integrity multiple times and by multiple people to ensure no mistakes were made in the evaluation of the accuracy. All the information provided in this research has been based on cited sources and verified information.

References

[1] Karan Aggarwal **andothers**. **?**Has the future started? The current growth of artificial intelligence, machine learning, and deep learning**? in***Iraqi Journal for Computer Science and Mathematics*: 3.1 (2022), **pages** 115–123.

[2] Wojciech Bieniecki, Szymon Grabowski **and** Wojciech Rozenberg. **?**Image preprocessing for improving ocr accuracy**? in***2007 international conference on perspective technologies and methods in MEMS design*: IEEE. 2007, **pages** 75–80.

[3] Cem Dilmegani. *Current state of OCR in 2023: Is it dead or a solved problem?* **may** 2020. URL: https://research.aimultiple.com/ocr-technology/#:~:text=of%5C%20OCR%5C%20tools?-,OCR%5C%20is%5C%20not%5C%20a%5C%20stand-alone%5C%20solution%5C%20in%5C%20human-machine, structured%5C%20data%5C%20from%5C%20their%5C%20documents..

[4] *Google Cloud, Vision API Pricing*. https://cloud.google.com/vision/pricing.

[5] Anh Khoi Ngo Ho, Jean-Christophe Burie **and** Jean-Marc Ogier. **?**Panel and speech balloon extraction from comic books**? in***2012 10th IAPR international workshop on document analysis systems*: IEEE. 2012, **pages** 424–428.

[6] Rose Holley. **?**How good can it get? Analysing and improving OCR accuracy in large scale historic newspaper digitisation programs**? in***D-Lib Magazine*: 15.3/4 (2009).

[7] Abhinav Jain **andothers**. **?**Overview and importance of data quality for machine learning tasks**? in***Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*: 2020, **pages** 3561–3562.

[8] *Kumiko, the Comics Cutter*. https://github.com/njean42/kumiko/.

[9] Pengyuan Lyu **andothers**. **?**Multi-oriented scene text detection via corner localization and region segmentation**? in***Proceedings of the IEEE conference on computer vision and pattern recognition*: 2018, **pages** 7553–7563.

[10] Maryam M Najafabadi **andothers**. **?**Deep learning applications and challenges in big data analytics**? in***Journal of big data*: 2.1 (2015), **pages** 1–21.

[11] Hong-Wei Ng **and** Stefan Winkler. **?**A data-driven approach to cleaning large face datasets**? in***2014 IEEE international conference on image processing (ICIP)*: IEEE. 2014, **pages** 343–347.

[12] Toru Ogawa **andothers**. **?**Object detection for comics using manga109 annotations**? in***arXiv preprint arXiv:1803.08670*: (2018).

[13] Wanli Ouyang **andothers**. **?**Factors in finetuning deep model for object detection with long-tail distribution**? in***Proceedings of the IEEE conference on computer vision and pattern recognition*: 2016, **pages** 864–873.

[14] Xufang Pang **andothers**. **?**A robust panel extraction method for manga**? in***Proceedings of the 22nd ACM international conference on Multimedia*: 2014, **pages** 1125–1128.

[15] *PHD Comics*. http://phdcomics.com/..

[16] Christophe Ponsard, Ravi Ramdoyal **and** Daniel Dziamski. **?**An ocr-enabled digital comic books viewer**? in***Computers Helping People with Special Needs: 13th International Conference, ICCHP 2012, Linz, Austria, July 11-13, 2012, Proceedings, Part I 13*: Springer. 2012, **pages** 471–478.

[17] Christophe Rigaud **andothers**. **?**Toward speech text recognition for comic books**? in***Proceedings of the 1st International Workshop on coMics ANalysis, Processing and Understanding*: 2016, **pages** 1–6.

[18] Yuji Roh, Geon Heo **and** Steven Euijong Whang. **?**A survey on data collection for machine learning: a big data-ai integration perspective**? in***IEEE Transactions on Knowledge and Data Engineering*: 33.4 (2019), **pages** 1328–1347.

[19] Salma **andothers**. **?**Development of ANPR framework for Pakistani vehicle number plates using object detection and OCR**? in***Complexity*: 2021 (2021), **pages** 1–14.

[20] Hillary Sanders **and** Joshua Saxe. **?**Garbage in, garbage out: how purportedly great ML models can be screwed up by bad data**? in***Proceedings of Blackhat*: 2017 (2017).

[21] Maciej Styczen. **?**Automated Text-Image Comic Dataset Construction**? in***Delft, the Netherlands: Delft University of Technology*: (2021).

7 Abstract

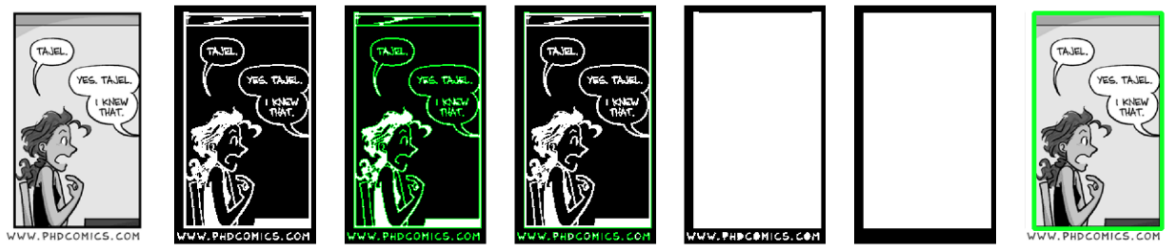


Figure 8: A visualization of the steps of the panel extraction procedure, for simplicity, presented on a single panel example. The same steps apply to a strip with multiple panels. From the left: a): the original image in grayscale, b): binarized image c): all the contours identified in the image, d): the outermost contours, e): the outermost contours filled with white color, f): noise removal using morphological opening g): proposed panel bounding box. (Comic strips from "Piled Higher and Deeper" by Jorge Cham www.phdcomics.com [15])

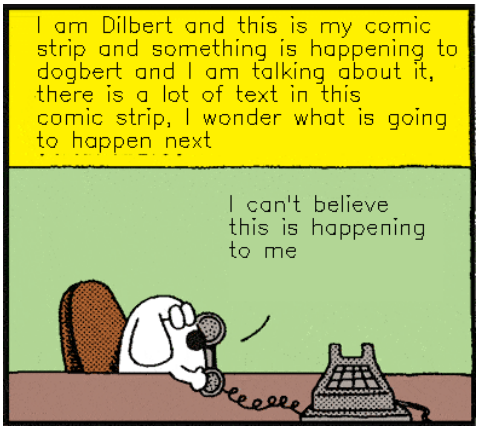


Figure 9: Editing panel with non-text bubble

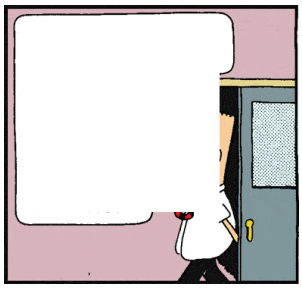


Figure 10: Editing panel with non-text bubble

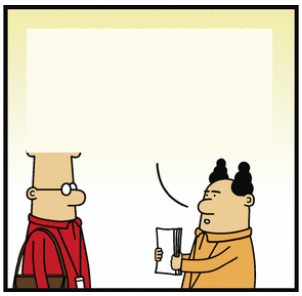


Figure 11: Editing panel with non-text bubble

Figure 12: Object detection model not finding "AAAGH!"

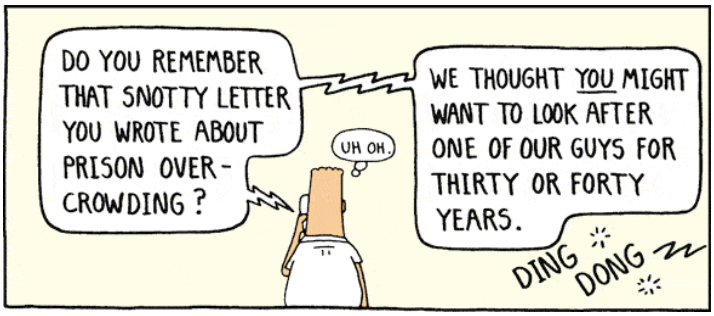
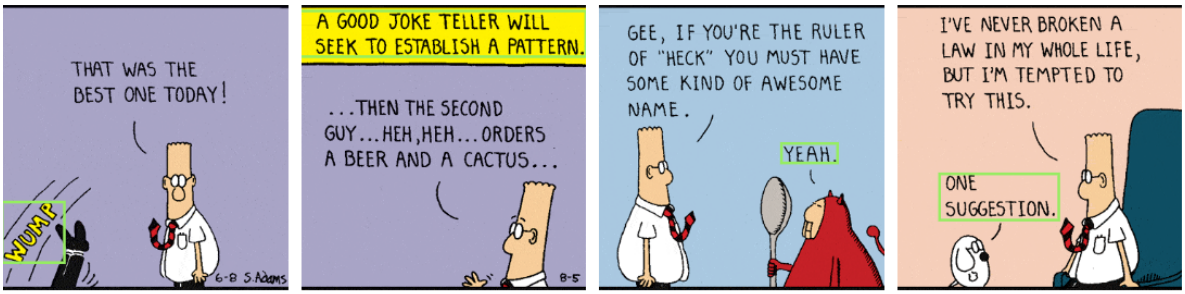
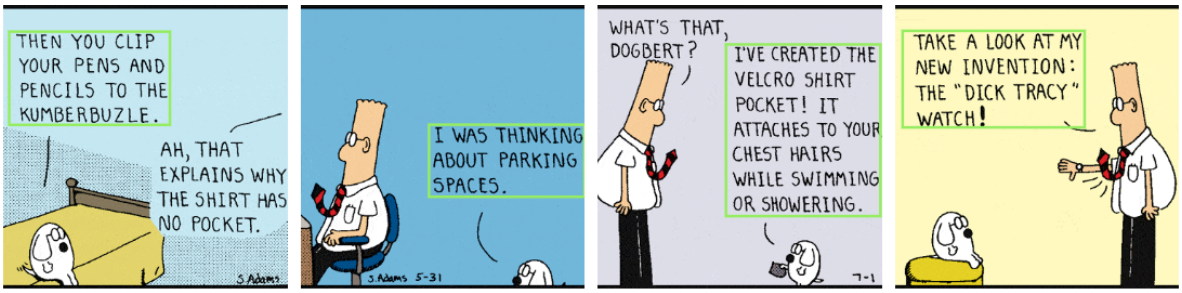


Figure 13: Object detection model not finding "UH OH" and "DING DONG"



Score: 0.718 Score: 0.889 Score: 0.925 Score: 0.925

Figure 14: Low confidence results of object detection (Green box highlights which text box is associated with the score)



Score: 0.994 Score: 0.995 Score: 0.996 Score: 0.996

Figure 15: High confidence results of object detection (Green box highlights which text box is associated with the score)

Figure 16: Editing and Deleting a comic panel

Figure 17: Editing and Deleting a comic panel