THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Extracting Information from Scientific Figures Using a Natural Language Processing Technique

PHAKPHUM ARTKAEW
SPRING 2023

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Computer Science
with honors in Computer Science

Reviewed and approved* by the following:

Ting-Hao ‘Kenneth’ Huang
Assistant Professor of IST
Thesis Supervisor

Ting He
Associate Professor of EECS
Honors Adviser

* Electronic approvals are on file.

ABSTRACT

Figures present an essential role in scientific papers as they convey critical messages to the audiences; however, low-quality captions frequently appear in scientific papers, which can create misunderstanding among readers and some trouble understanding. Natural language processing can help generate a concise and clear caption to improve a better understanding of scientific captions. In this thesis, we aim to create a baseline to test whether the existing natural processing language model can outperform the heuristic method of creating captions; such a method is to get the information from the X-Y graph using the EasyOCR and put them into our 45 caption templates. We use the SCICAP dataset as the main dataset to extract X-Y axis information and create the caption template. The caption template is created by going through the SCICAP dataset and writing down the common pattern that can be found. As a result, the dataset of captions is created and can be used as a baseline to compare with other machine generated-text by using human evaluation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my thesis supervisor, Professor Kenneth Huang for the opportunity to work and conduct research in CrowdAI. Being in this lab has given me much research experience in the natural language processing field which I could not find anywhere. I would also like to thank the member of the lab who always guided me whenever I had doubts especially Chieh-Yang Huang and Ting-Yao Hsu. They taught me a lot about natural language processing and how to use the Penn State Linux server which I struggled with it a lot in the beginning. Their insights and feedback have been invaluable in shaping my ideas and helping me to refine my work. This has been a fun experience for me and would love to conduct this kind of research again in the future.

In addition, I would like to thank my co-researcher, Reuben Lee, for being such an incredible friend. I thank you for your encouragement and support throughout the project.

Lastly, I would like to thank my lovely girlfriend, Khim, who always loves and supports me; I am blessed to have you in my life.

Thank you all for your contributions and support.

Chapter 1

Introduction

Scientists use scientific papers to convey and communicate important messages to other researchers. Each paper can contain various elements such as text, table, or figure; each helps illustrate the paper's main idea. The text occupies the majority of a paper and is the main content, while tables and figures are there to help readers see a clearer picture. Readers can get information from tables easily since it is defined by rows and columns. However, understanding the figure clearly requires a deep understanding of the context and its information on the X-Y axis. To help navigate the idea, researchers always put captions under each figure, but sometimes they make it harder to comprehend and may decrease the paper's overall understanding. To improve the caption’s quality, researchers have been trying to use various techniques such as natural language processing to re-generate captions; this introduces a new research area—figure caption generation Various approaches have been made to this particular problem, but some researchers say to look at the figure first and then use some tool and techniques to generates the caption; some says to look at their mentions in the paper. However, the question of how good the generated captions remain unanswered.

Chapter 2

Background

SCICAP

SciCap is the dataset used in this thesis. It was created by a team of researchers from Pennsylvania State University led by Ting-Yao Hsu in his experiment on figure caption generation. The dataset, called SciCap, contains more than 416,000 figures and their captions; this was extracted from over 290,000 computer science papers published between 2010 and 2020 from arVix. More detail about the dataset can be found in his paper. The dataset itself is huge; it is about 18.15 GB. The organization of the dataset can be found below (Figure 1).

```
scicap_data.zip
├── SciCap-Caption-All                #caption text for all figures
│       ├── Train
│       ├── Val
│       └── Test
├── SciCap-No-Subfig-Img              #image files for the figures without subfigures
│       ├── Train
│       ├── Val
│       └── Test
├── SciCap-Yes-Subfig-Img             #image files for the figures with subfigures
│       ├── Train
│       ├── Val
│       └── Test
├── arxiv-metadata-oai-snapshot.json  #arXiv paper's metadata (from arXiv dataset)
└── List-of-Files-for-Each-Experiments  #list of figure names used in each experiment
    ├── Single-Sentence-Caption
    │   ├── No-Subfig
    │   │   ├── Train
    │   │   ├── Val
    │   │   └── Test
    │   └── Yes-Subfig
    │       ├── Train
    │       ├── Val
    │       └── Test
    ├── First-Sentence                #Same as in Single-Sentence-Caption
    └── Caption-No-More-Than-100-Tokens #Same as in Single-Sentence-Caption
```

Figure 1. The organization of SciCap dataset

All files in SciCap-Caption-ALL, SciCap-No-Subfig-Img, and SciCap-Yes-Subfig-Img contain figures in the following format: [arVix paper id]-[figure-id].png format. the snapshot file is a

jsonl file that contains the following ten columns: paper-ID, figure-ID, figure_index, paragraph, mentions, mention_sentence_index, caption, Img_text, caption_no_index, and 1st_caption.

EasyOCR

EasyOCR is an open-source Python-based optical character recognition that is designed to be easy to use and highly accurate. It is a library that can be integrated into Python applications to extract text from images and PDF files. This tool uses deep learning algorithms and pre-trained models to recognize text in more than 70 languages. Those languages include but are not limited to the following: English, Chinese, Japanese, Korean, and many others. It can also recognize a wide range of fonts, including handwritten text.

Installation

we can install EasyOCR by using pip install easyocr command. For more information about installation, you can consult their GitHub.

EasyOCR is easy to use; here below is how to use EasyOCR on one of the figures.

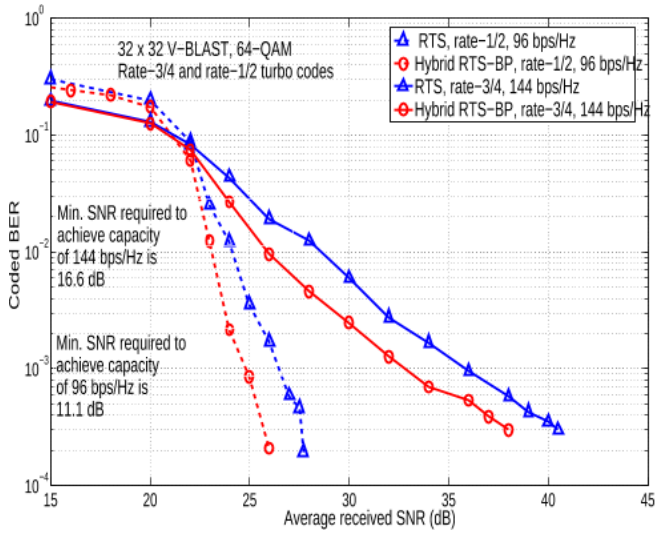

Figure 2. Scicap Figure

```
  4 import easyocr
  3 import cv2
  2
  1 reader = easyocr.Reader(['en'])
5   img = cv2.imread('/Users/phakphumartkaew/Documents/crowdAI/scicap_data/SciCap-No-Subfig-Img/val/1001.2376v1-Figure5-1.png')
  1 result = reader.readtext(img,rotation_info=[90,180,270])
  2 for i in result:
  3     print(i)
```

Figure 3. example code to run easyOCR

```
  warnings.warn(msg)
([[305, 19], [429, 19], [429, 37], [305, 37]], 'RTS, rate-2, 96 bpsiHz', 0.49007301463894504)
([[81, 25], [217, 25], [217, 43], [81, 43]], '1', 0.9240319428318458)
([[305, 35], [473, 35], [473, 55], [305, 55]], 'Hybrid RTS-BP, rate-12, 96 bpslH', 0.5777191751877104)
([[81, 41], [247, 41], [247, 59], [81, 59]], 'Rate-3/4 and rate-1/2 turbo codes', 0.6255815433796097)
([[303, 53], [431, 53], [431, 71], [303, 71]], 'RTS, rate-314,144 bpsIHz', 0.5906535812266791)
([[303, 69], [469, 69], [469, 89], [303, 89]], '1', 0.22960825353459668)
([[35, 149], [137, 149], [137, 165], [35, 165]], 'Min, SNR required to', 0.725042664347954)
([[35, 167], [117, 167], [117, 183], [35, 183]], 'achieve capacity', 0.9333753998371358)
([[0, 180], [116, 180], [116, 204], [0, 204]], '1', 0.3360658341989655)
([[33, 197], [75, 197], [75, 215], [33, 215]], '16.6 @B', 0.681885144332386)
([[0, 202], [8, 202], [8, 228], [0, 228]], '1', 0.44705213355422657)
([[33, 247], [137, 247], [137, 263], [33, 263]], 'Min; SNR required t0', 0.7503550465853243)
([[33, 265], [117, 265], [117, 281], [33, 281]], 'achieve capaciv:', 0.7048893562494248)
([[35, 281], [107, 281], [107, 297], [35, 297]], 'of 96 bpsIHz is', 0.4947847778966597)
([[33, 295], [75, 295], [75, 313], [33, 313]], '11,1dB', 0.9178436229259817)
([[205, 379], [335, 379], [335, 395], [205, 395]], 'Average received SNR (B)', 0.502095389391633)
```

Figure 4. example output of easyOCR

The output consists of the list of words boundary, the words themselves, and the confidence rate.

We note that the x-y axis for positioning in easyOCR looks like the figure below. The smallest x at the leftmost then increases as we go right; the smallest y at the leftmost then increases as we go down.

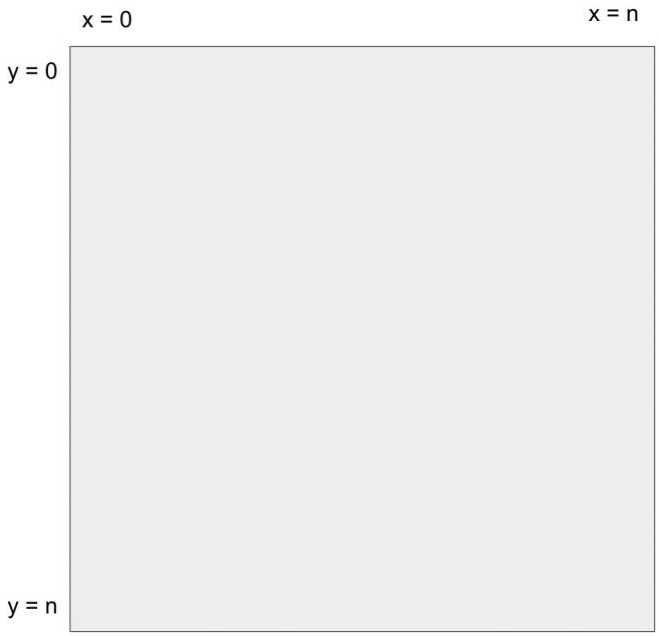

Figure 5. easyOCR positioning system

Chapter 3

Methodology

Approach

Many researchers purpose many ways of generating captions from figures and those can give different results, but their performance is still questionable. We believe humans or readers are the best judges to evaluate the generated captions' performance; they are the ones who read the paper. We purpose a simple baseline to classify good and bad-generated captions; such a baseline is created using a heuristic method that we believe a caption is simply the x-axis versus the y-axis. Therefore, if we say that a heuristic method is a good enough or fair caption for readers to understand, a higher technique should give a more comprehensive caption; we can conduct an experiment and let readers choose which captions are easier for them to understand. The workflow of this work can be illustrated in the figure below.

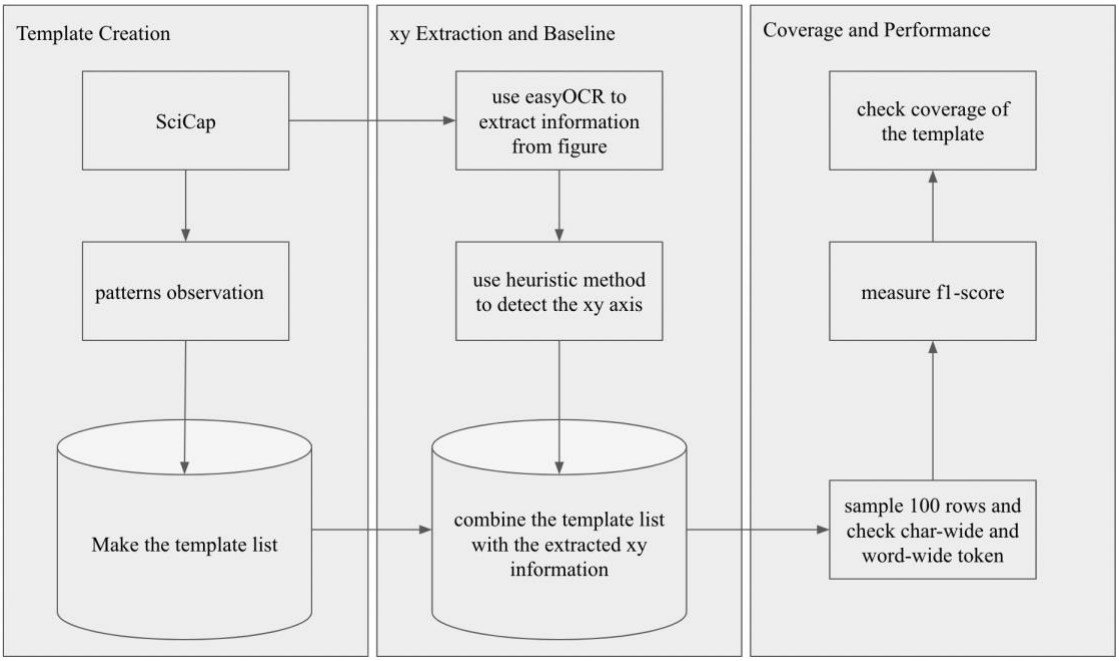

Figure 6. Workflow of the thesis

Caption pattern gathering

There can be various versions of the simple caption such as [Y-axis] vs [X-axis], [Y-axis] of data with the [X-axis], and many more. The baseline needs to cover some of the variety to give the user more choices to choose between the baseline and the generated captions. Therefore, the first step was to gather information on how the pattern of captions is, then, store it in the template list. We obtained 45 patterns below by observing the patterns in the snapshot JSON file.

row	**Caption**
1	[Y-axis] vs [X-axis]
2	[Y-axis] v [X-axis]
3	[Y-axis] obtained by varying the [X-axis]
4	Trade-off between [X-axis] and [Y-axis]
5	A plot of [Y-axis] for each [X-axis] value
6	[Y-axis] of data with the [X-axis]
7	[Y-axis] for the [X-axis]
8	[Y-axis] varying the [X-axis]
9	[Y-axis] as a function of [X-axis]
10	[Y-axis] and [X-axis]
11	[Y-axis] of regulation variants over [X-axis]
12	[Y-axis]-[X-axis] curves
13	Comparison between [Y-axis] and [X-axis]
14	Comparison between the performance of [X-axis]

15	Relationship between [Y-axis] and [X-axis]
16	[Y-axis] of [X-axis]
17	Variation of [Y-axis] with [X-axis]
18	[Y-axis] for the [X-axis]
19	graph of [Y-axis] by [X-axis]
20	[Y-axis] distribution of [X-axis]
21	The transformer [Y-axis] vs number of [X-axis]
22	[Y-axis] results obtained on … using … from the [X-axis]
23	[Y-axis] with [X-axis]
24	Tracking [Y-axis] over [X-axis]
25	Validation [Y-axis] and after each [X-axis]
26	[Y-axis] of [X-axis]
27	[Y-axis] w.r.t. varying [X-axis]
28	[Y-axis] of return prediction by using [X-axis]
29	A plot of the relative [Y-axis] achieved for various [X-axis]
30	demonstrated by [Y-axis] using [X-axis]
31	[Y-axis] evolution with the increasing number of [X-axis]
32	[Y-axis] of the model w.r.t. to [X-axis]
33	[Y-axis] score on a different group of [X-axis]
34	[Y-axis] at different [X-axis]
35	[Y-axis] values for different [X-axis]
36	Effect of [X-axis] on [Y-axis] for the

37	the influence of [X-axis] to the [Y-axis]
38	Distribution of the [Y-axis]
39	Comparison of [Y-axis] on [X-axis]
40	Correlation between the [X-axis] and [Y-axis]
41	The figure shows the relative [Y-axis] for [X-axis]
42	[Y-axis] for … after each [X-axis]
43	[Y-axis] between representations at each [X-axis]
44	A plot of [Y-axis] as a function of [X-axis]
45	Variation of [Y-axis] with respect to [X-axis]

Table 1. patterns we observed from SciCap

[Y-axis] will be replaced by the y-axis found in the figure and [X-axis] will be replaced by the x-axis found in the figure.

XY axis extracting information

SciCap contains figures in a .png image file and in order to extract the x-y information from the figure, we use the easyOCR tool to do it. Generally, the figure in SciCap looks like the figure below which shows the x-axis in horizontal text and the y-axis in vertical text.

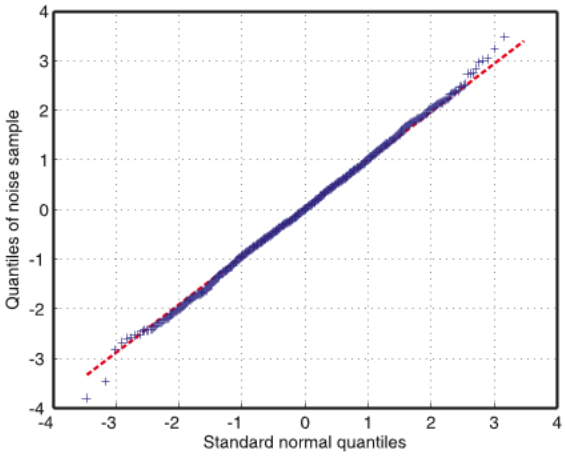

Figure 7. A normal SciCap figure

Although easyOCR can detect rotation text like the y-axis, we found it better to rotate the figure image and use easyOCR to detect text again just like the horizontal one. An example of a rotated figure is shown below.

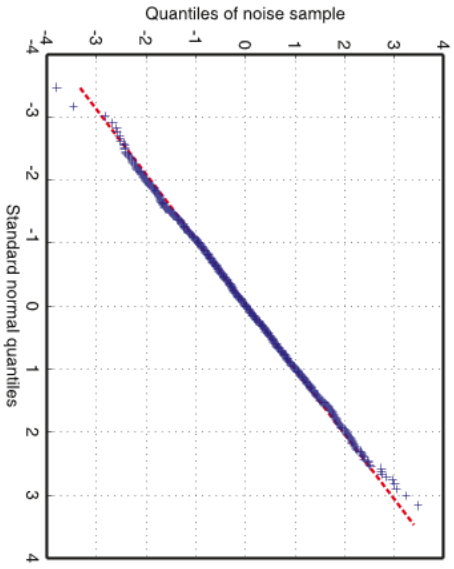

Figure 8. A rotated figure

The code below is used to detect all texts in the figure and create a new dataset for us to use in creating the baseline.

phakphumartkaew — p

```
import os
import easyocr
import cv2
import pandas as pd
from tqdm import tqdm
path1 = "SciCap-No-Subfig-Img/val"
path2 = "SciCap-No-Subfig-Img/val/rotate"

scicapNoSubfigVal = os.listdir(path1)
scicapNoSubfigValRotate = os.listdir(path2)
reader = easyocr.Reader(['en'])
sz = len(scicapNoSubfigValRotate)
print(sz)
results=[]
for i in tqdm(range(sz)):
    if scicapNoSubfigValRotate[i]=='rotate':
        continue
    img = cv2.imread(path2+'/'+scicapNoSubfigValRotate[i])
    toappend = []
    toappend.append(scicapNoSubfigValRotate[i])
    result = reader.readtext(img,rotation_info=[90,180,270])
    toappend.append(result)
    results.append(toappend)
df = pd.DataFrame(results,columns=['figure-ID','Img_txt'])
print(df.size)
df.to_json('captionValParameterRotate.json')
```

Figure 9. Code of extract information from SciCap figures

Explanation:

We have two folders for the regular figures and the rotated one. Then, we loop through every figure in the particular folder and append the result to the results list. After that, we can export the data into a JSON file noting that we export two pieces of information: the figure id and its information. We have to run this code on all the test, train, and val folder; also for the regular figures and the rotated figures.

```
(envo) python example.py
<bound method NDFrame.head of                          figure-ID                                            Img_txt
0      1907.04292v1-Figure5-1.png  [[[[2, 92], [10, 92], [10, 134], [2, 134]], St...
1      2007.14680v1-Figure5-1.png  [[[[265, 7], [516, 7], [516, 33], [265, 33]], ...
2      1909.08148v1-Figure8-1.png  [[[[356.2567058538, 13.3310352684], [393.89950...
3      1911.08145v3-Figure2-1.png  [[[[81, 15], [161, 15], [161, 31], [81, 31]], ...
4      2011.03143v1-Figure4-1.png  [[[[294, 34], [424, 34], [424, 60], [294, 60]]...
...                           ...                                                ...
13349  2010.07359v1-Figure6-1.png  [[[[269, 63], [393, 63], [393, 77], [269, 77]]...
13350   1104.0752v1-Figure1-1.png                                                 []
13351  1904.00800v2-Figure3-1.png  [[[[156, 0], [418, 0], [418, 22], [156, 22]], ...
13352  2008.06992v1-Figure2-1.png  [[[[29, 61], [57, 61], [57, 79], [29, 79]], 25...
13353  1701.09094v1-Figure3-1.png  [[[[245, 31], [335, 31], [335, 47], [245, 47]]...
```

Figure 10. example output after extracting information from SciCap figures

creating a baseline

Since we have all the information of figures in a JSON file, we can use it to extract the x-axis and y-axis information. We observe that the x-axis will often be the bottommost text in the figure, while the y-axis will often be the leftmost text. We note that this is not always the case but it is good enough for us to extract the xy axis and use it to construct the baseline. The code below is used to create the baseline

```python
import csv
import pandas as pd
from tqdm import tqdm

captionTrain = pd.read_json('captionValParameter.json')
captionTrainRotate = pd.read_json('captionValParameterRotate.json')

lenFigure = len(captionTrain['figure-ID'])
bigDict={}
print(lenFigure)
#loop through normal img to select the x-axis
#or the bottommost one
#which will be the most top one from the graph data (lagrest y)
for i in tqdm(range(lenFigure)):
    figureID = captionTrain['figure-ID'][i]
    if(figureID not in bigDict):
        bigDict[figureID] = {}
    lenImg = len(captionTrain['Img_txt'][i])
    bottom=0
    bottomTxt=""
    for j in range(lenImg):
        lenBox = len(captionTrain['Img_txt'][i][j][0])
        for k in range(lenBox):
            if(captionTrain['Img_txt'][i][j][0][k][1]>bottom):
                bottom = captionTrain['Img_txt'][i][j][0][k][1]
                bottomTxt = captionTrain['Img_txt'][i][j][1]
    bigDict[figureID]['x-axis']=bottomTxt

#loop through rotated img to select the y-axis
```

Figure 11. Part one: get the bottommost text

First, we read the two files which contain all information in normal figures and rotated figures. In normal figures, the x-axis will be at the bottommost which can be interpreted as the largest y-value. This code simply remembers the new largest value as it goes through the dataset, then, appends the bottommost text as the x-axis to its corresponding figure-id.

```
28
29 #loop through rotated img to select the y-axis
30 #or the leftmost one
31 #which will be the most bottom y(smallest y)
32 lenFigureRotate = len(captionTrainRotate['figure-ID'])
33 print(lenFigureRotate)
34 for i in tqdm(range(lenFigureRotate)):
35     figureID = captionTrainRotate['figure-ID'][i]
36     lenImg = len(captionTrainRotate['Img_txt'][i])
37     top = 1000000
38     topTxt=""
39     for j in range(lenImg):
40         lenBox = len(captionTrainRotate['Img_txt'][i][j][0])
41         for k in range(lenBox):
42             if(captionTrainRotate['Img_txt'][i][j][0][k][1]<top):
43                 top = captionTrainRotate['Img_txt'][i][j][0][k][1]
44                 topTxt = captionTrainRotate['Img_txt'][i][j][1]
45     bigDict[figureID]['y-axis']=topTxt
46
```

Figure 12. Part two: get the top text

Because we rotate figures 90 degrees clockwise to make the vertical text becomes readable to the OCR tool, the y-axis will be at the very top of the figure. In other words, the y-axis is the text where the y-value is the smallest. This code loops through the dataset and keeps the new lowest y-value and its text. Then, we append this text as a y-axis to its corresponding figure id.

```
 92
 93 for i in bigDict:
 94     print(i)
 95     xAxis = bigDict[i]['x-axis']
 96     yAxis = bigDict[i]['y-axis']
 97     bigDict[i]['template']=[]
 98     for j in template:
 99         txt = j.replace('[X-axis]',xAxis)
100         txt = txt.replace('[Y-axis]',yAxis)
101         bigDict[i]['template'].append(txt)
102 #extract it to the template
103 df = pd.DataFrame.from_dict(bigDict)
104 df.to_json('xyVal.json')
```

Figure 13. Part three: combine those two into the template and extract them to a JSON file.

After we have all the x-y axis for each figure id, we can extract them into a JSON file using Pandas Dataframe. We repeated this code for the train, test, val folder to create a baseline for each one.

Coverage and similarity

```
#finding top five
figcapData = pd.read_json('fig_idx_match_paragraph_single_graphplot(update).jsonl',lines=True)
similarity = pd.read_csv('similarity.csv')
#create list of tuples as a placeholder (template,value similarity)
new_similarity = []
#print(len(similarity['columns']))
#print(len(template))
for i in range(len(similarity['columns'])):
    #new_tup=tuple((template[i],similarity['columns'][i]))
    new_tup=tuple((similarity['columns'][i],template[i]))
    new_similarity.append(new_tup)
#sort it to find the top five
new_similarity.sort(reverse=True)
df = pd.DataFrame(new_similarity)
```

Figure 14. reading files

```
"""

#captionPath = "/home/pma5244/workspace/scicap_data/SciCap-Caption-All"
tokensFromFindings=set()
for i in range(len(new_similarity)):
    token = new_similarity[i][1].split()
    for j in token:
        tokensFromFindings.add(j)
    if(i==4):
        #choose only top five
        break
#test = os.listdir(captionPath+"/test")
#train = os.listdir(captionPath+"/train")
#val = os.listdir(captionPath+"/val")

allTokens = 0
overlappedToken = 0

for i in tqdm(range(len(figcapData))):
    caption = figcapData['caption'][i]
    token = caption.split()
    for j in token:
        allTokens+=1
        if(j in tokensFromFindings):
            overlappedToken+=1

print(overlappedToken)
print(allTokens)
```

Figure 15. finding tokens

The coverage of our template can be found by counting the overlapped token and all tokens. We made our template into unique tokens and loop through the dataset to count the overlapped ones. Moreover, we used spaCy to measure the similarity score for each template to see how much the top five can cover. This was done by looping through the dataset and the template to calculate the score for each one of them. Then, add the score to its corresponding template. After finishing,

we can get a sense of which template is similar to the captions in the SciCap dataset. The code below shows the procedure for calculating the similarity score and its output.

```
"""
similarity = [0.0]*len(template)
for i in tqdm(range(len(figcapData))):
    caption = nlp(figcapData['caption'][i])
    for j in range(len(template)):
        similarity[j]+=(nlp(template[j]).similarity(caption))
df = pd.DataFrame(similarity,columns=["columns"])
df.to_csv('similarity.csv',index=False)
"""
```

Figure 16. calculating similarity code

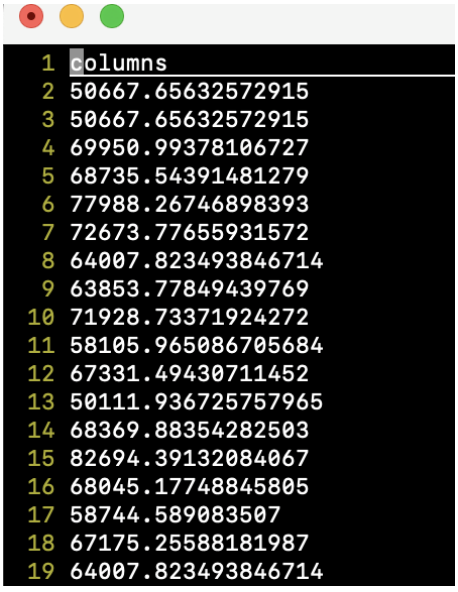

Figure 17. example output of the similarity score

Chapter 4

Result and Analysis

After we obtained the baseline from running the baseline code on the train, test , val dataset, now we have three baselines to compare the performance with the machine-generated captions. Here below shows the first three captions in the train baseline; it consists of the x-axis, the y-axis, and the template captions after applying to the xy information.

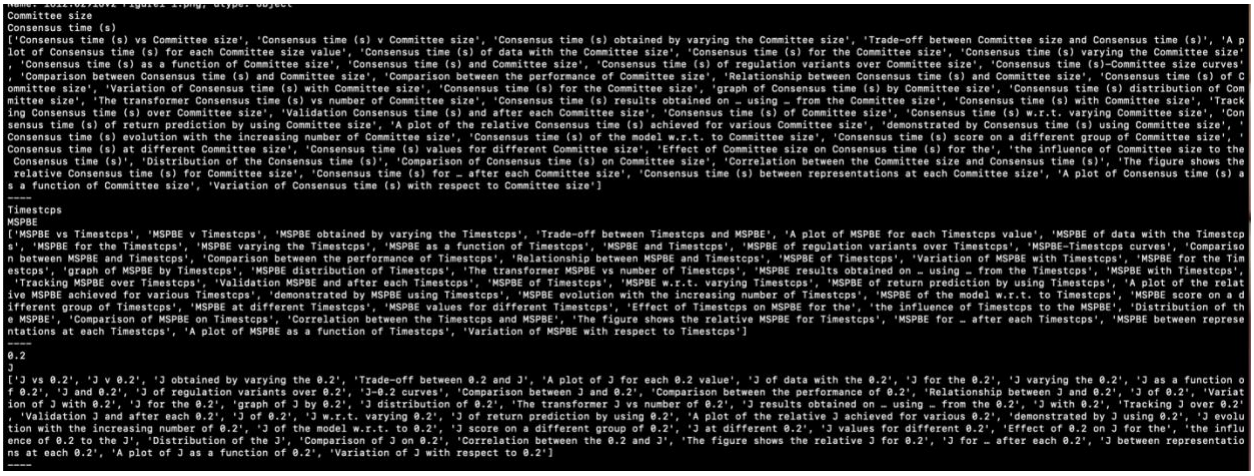

Figure 18. example output of the baseline

Performance of xy extraction

In the extraction, we assume that the x-axis will be the bottommost axis and the y-axis will be the leftmost axis. This method is not perfect and there exist errors in the extraction like the third column shown above. Our templates cover about 26% of the captions variety in the scicap dataset and its top five patterns cover about 16%. To test the accuracy, we utilized f1-score testing which measures the recall and precision of word-wide tokens and char-wide tokens.

We sampled 100 rows from the extraction and measured their precision and recall by hand which can be found in the table below.

Word-wide	Precision	0.6820702403
	Recall	0.7639751553
	F1	0.720703125
Char-wide	Precision	0.8103273709
	Recall	0.9118875807
	F1	0.8581129378

Table 2. The extraction performance

On average, the f1-score range is 0.7, and the good enough score should be above 0.5. Therefore, we interpret that our extraction method is good enough.

Coverage

All	Overlapped tokens	715804
	All tokens	2702129
	Percent covered	26%
Top five	Overlapped tokens	453625
	All tokens	2702129
	Percent covered	16%

Table 3. The coverage performance

Our template does not cover most of the caption patterns because we labeled them by hand. Further work is needed to find a better way to label patterns.

Chapter 5

Conclusion and Future Work

Figure caption generation is a relatively new research area and computer scientists are looking for an opportunity for computers to help write better captions. To help measure the performance of computer-generated captions, we purpose a simple baseline that extracted the xy-information and apply them to the patterns we observed in the SciCap dataset. As a result, our performance of extracting the xy-information is 0.7 which performs better than the good enough score. However, due to limited time, we did not conduct an experiment to test our baseline to other caption-generation methods. In the future, we plan to conduct a human experiment to compare our baseline with other computer-generated captions to test whether other methods create good enough captions. The experiment can be conducted on sample data of SciCap and compared with the same captions from our baseline by letting users choose the better captions. After that, we can measure the statistics and conclude whether such methods perform better than our baseline.

Appendix A

SCICAP github repo: https://github.com/tingyaohsu/SciCap

F1 score checking:

https://docs.google.com/spreadsheets/d/1FSHrvDpABXy7ecXk894dCa42Qo_KZMvI7oS5lso7Lzk/edit?usp=sharing

All the code and CSV files and be found here:

https://drive.google.com/drive/folders/1yXVHlQfFL6IF4dIhWnIfeNX9VT911oJT?usp=sharing

BIBLIOGRAPHY

Hsu, T.-Y., Giles, C. L., & Huang, T.-H. (2021). SciCap: Generating captions for scientific figures. Findings of the Association for Computational Linguistics: EMNLP 2021. https://doi.org/10.18653/v1/2021.findings-emnlp.277

Huang, C., Hsu, T., Rossi, R., Nenkova, A., Kim, S., Chan, G. Y., Koh, E., Giles, C. L., & Huang, T. '. (2023). Summaries as captions: Generating figure captions for scientific documents with automated text summarization.https://doi.org/10.48550/arxiv.2302.12324

JaidedAI. (n.d.). *Jaidedai/EasyOCR: Ready-to-use OCR with 80+ supported languages and all popular writing scripts including Latin, Chinese, Arabic, Devanagari, cyrillic and etc.*. GitHub. Retrieved April 3, 2023, from https://github.com/JaidedAI/EasyOCR

Tokenization. (n.d.). Retrieved April 3, 2023, from https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html

Wikimedia Foundation. (2023, February 17). *F-score*. Wikipedia. Retrieved April 3, 2023, from https://en.wikipedia.org/wiki/F-score

Wikimedia Foundation. (2023, February 7). *Heuristic*. Wikipedia. Retrieved April 3, 2023, from https://en.wikipedia.org/wiki/Heuristic

ACADEMIC VITA

Phakphum Artkaew

www.linkedin.com/in/phakphum-artkaew

EDUCATION

The Pennsylvania State University, Schreyer Honors College — **University Park, PA**
College of Engineering | Bachelor of Science in Computer Science — *Class of 2023*

RELEVANT EXPERIENCE

Crowd-AI Lab — **University Park, PA**
Undergradute Researcher — *Oct. 2021 - Present*

- Research NLP and assist graduate student in figure caption generation project
- Improved and utilized equation detection regular expression to 200,000 scientific figures
- Used easyOCR to detect texts in figures and create x-y templates from detected texts
- Research on logical fallacies detection on social media

Intelligent Vehicles and Systems Group — **University Park, PA**
Research Assistant — *June. 2022 – Aug. 2022*

- Facilitated the PennDOT-funded project, *Safe Integration of Automated Vehicles into WorkZones*
- Implemented test track into OpenDrive xodr file

Agoda — **Bangkok, Thailand**
Software Engineer Intern — *May 2021 – Aug. 2021*

- Designed and implemented content back office tool for generating ,viewing, deleting, resetting caches on property content using **Vue.js**
- Implemented API endpoints for connecting to **Couchbase** using **Scala**
- Utilized **Kafka** producer and consumer for deleting and resetting caches

Mahfuza Farooque Lab — **University Park, PA**
Undergraduate Researcher — *Mar. 2020 – Feb. 2021*

- Created text-to-text machine learning models to detect nine allergens in different kinds of food
- Collected public datasets from Kaggle, cleaned dataset, and performed **LinearSVC** on it
- Selected to present the work at 2021 IEEE North American Virtual International Student Conference

EXTRA ACTIVITIES

Penn State Association for Computing Machinery — **University Park, PA**
Treasurer — *Dec. 2020 – Dec. 2022*

- Handle club's financial activities
- Organize annual university competitive programming competition

AlgoPSU Captain — *Aug. 2022 –Dec. 2022*

- Organize weekly meeting to help students apply their data structure and algorithms knowledge through competitive programming

Otog.cf — **Khon Kean, Thailand**
Administrator — *May 2017 – Dec. 2022*

- Oversee programming grader website to facilitate underclassmen's learning of competitive programming

Thai Student Association at Penn State — **University Park, PA**
Media and Communication Officer — *May 2020 – May 2021*

- Created event posters and infographics for club
- Posted news and events on club's Instagram and Facebook

Penn State Advanced Vehicle Team (ADAS) **University Park, PA**
Volunteer *Aug. 2019 – Dec. 2019*

- Volunteered for the ADAS team to help create a self-driving car
- Prepared working areas to make sure it is ready to use

AWARDS AND HONORS

17th place, ACM ICPC ECNA	2019
Finalist, 13th Thailand Olympiad Informatics	2018
1st place, DevPSU Startup	2020
Thai Scholarship Recipient	2019