Utilizing Contextual Data to Improve Accessible Representations of Web Graphics

Namdar Nejad
namdar.nejad@mail.mcgill.ca
McGill University
Montréal, Québec, Canada

Supervised by Prof. Jeremy R. Cooperstock
Mentored by Juliette Regimbal and Rohan Akut

ABSTRACT

The Internet Multimodal Access to Graphical Exploration (IMAGE) project aims to produce accessible representations of web graphics using rich audio and touch. In this project, we use contextual data from the web page, starting with alt-text, to provide additional data to aid in rendering these representations. We create a program, instantiated in a microservice running on the IMAGE server, that will process the text content using Natural Language Processing (NLP) methods, namely Named Entity Recognition. In this communication, we describe our approach to creating and using the aforementioned program. We describe our implementation and how it can be used as a base for future contextual data preprocessors. Moreover, we explain how contextual data may be used to improve the overall IMAGE graphics rendering.

1 INTRODUCTION

The Internet Multimodal Access to Graphical Exploration (IMAGE) project at the McGill Shared Reality Lab (SRL) aims to improve the accessibility of web images and graphics by processing them to produce rich audio, haptic output and spoken text description on demand. The IMAGE project follows a three-step process consisting of data collection, preprocessing, and data synthesis to produce outputs describing the input graphics [1].

The current data preprocessors being used by the IMAGE project are primarily utilizing machine learning models that extract information directly from the graphic (e.g., colors, objects) rather than the contextual data (e.g., surrounding or alternative text). However, contextual data can be used to obtain information not present in the graphics. The inclusion of a contextual data-based preprocessor can aid in creating more targeted and accurate descriptions of graphics. For example, given an image of The Alps, as seen in Figure 1, the current graphics-focused preprocessors are able to describe the objects present in the image, e.g., sky, mountain, tree, and earth. This is done by applying various machine learning models on the image directly. Contextual data, on the other hand, can be used to describe the location where the photograph was taken, what specific mountains are in the images, etc.

Figure 1: An image from the Alps with the alt-text “The Eiger (shown along with the Mönch and the Jungfrau) has the tallest north face in the Alps.” [2]

In this project, we explore the potential of using contextual data, specifically alternative text (alt-text), as an additional preprocessing step in the IMAGE project. We introduce a preprocessor built on alternative text. The preprocessor extracts the alt-text and graphics from the raw input, evaluates its relevance to the image using a Reference-free Evaluation Metric, and employs the Stanford Named Entity Recognizer (NER) to label the Named Entities from the alt-text.

2 BACKGROUND

2.1 The IMAGE Project

Graphic materials such as charts and photographs on the internet still remain inaccessible to people with low vision. Although there are solutions for rendering graphical information, they are often limited to manually generated alt-text HTML labels, which are often erroneous and lacking in richness. The IMAGE project provides an alternative solution to rendering web graphics in which the visual experience is replaced with multimodal sensory feedback, rendered in a manner that helps overcome access barriers for blind and partially sighted users. The project is an open-source

code base that uses a Google extension to collect graphical data and artificial intelligence (AI) technology for the interpretation of maps, charts, and photographs [1].

The IMAGE project uses a server-client architecture and supports a three-step pipeline. The initial step is the data collection step that is done on the client side via a Google Chrome extension. The data collected is either an image file (including contextual data of the image), a geographic coordinate representing a point of interest displayed on an embedded Google map, or certain embedded charts. In the processing step, the data collected from the client at the previous step is passed on to microservices on the server to be preprocessed. These preprocessors receive HTTP POST requests containing the data and perform a specific task on them, which is typically extracting specific information and applying a machine learning model to them. The outputs of these preprocessors may be combined, allowing a preprocessor to take advantage of the output of any earlier stages [1].

Finally, at the synthesis stage, another set of microservices called handlers receives the entire set of information from the preprocessors and all information collected by the client. Each handler then determines if it is capable of producing a usable output from the given preprocessor and produces an output using built-in technologies and services that perform functions (e.g., text-to-speech) that are generic to multiple handlers. Multiple renderings for a single graphic may be produced by the handlers, providing different perspectives [1].

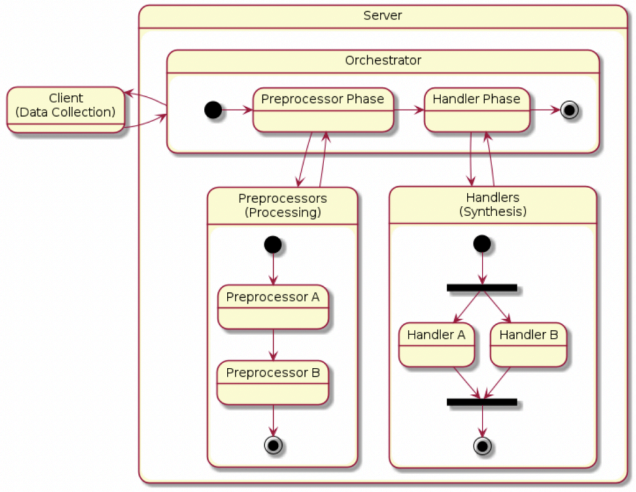

Figure 2: A high-level diagram of the software architecture showing how it models the three-step pipeline. [1]

2.2 Alternative Text

Contextual data, in the case of web graphics, provide background information, helping with a broader understanding of the graphic. Contextual data comes in many forms; for example, an article text, title, or the alt-text in the HTML of a given image in the article.

Alternative text (alt-text) is a descriptive text aimed to convey the meaning and context of a graphic and the primary source of providing visual information to the visually impaired. Although in some cases, alt-text may not be provided or may be uninformative, many popular web resources provide alt-text for their graphics, and the adaptation of web accessibility practices is on the rise. For example, Guinness et al. found that images that include alt-text on large commercial and news websites increased from approximately 40% in 2006 to 72% in 2018 [3]. However, in many cases, additional captioning was needed to allow for a better understanding of the content of the image [4].

We decided to use alt-text as the initial scope of the project as it provides a concise description of an image in, at most, a few sentences. The succinct nature of alt-texts enables us to apply NLP models to them and associate their information with the graphics more easily. While our approach in this project can also be applied to entire passages, with multiple paragraphs, it is harder to interpret the information gained.

Although contextual data is not always guaranteed, they can be valuable in interpreting the images that have them. The addition of contextual data preprocessors to the IMAGE source code can help in producing more accurate descriptions by providing information that can't be directly obtained from the graphics. In these cases, the information obtained from the alt-text may be used in the synthesis step in combination with the other preprocessors. For example, in the image in Figure 3, the IMAGE Chrome extension (version 0.4.0 used in the November of 2022) detects grass, trees, 4 people, and a dog; however, it's impossible to recognize the individuals in the image or the specific location the picture was taken from the graphics alone. The alt-text mentioning "*Michelle Obama and her oldest daughter walk a small black dog whilst Barack Obama, and their youngest daughter walks a short distance behind*" can be used to realize that two of the four recognized people are potentially Michelle and Barack Obama. Moreover, in Figure 4, the Chrome extension only recognizes the nature behind the dogs-recognizing, specifically recognizing "sky, rock, water, mountain, tree, and 2 dogs", while the exact location the photograph was taken can be obtained from the alt-text.

Moreover, the information gained from the contextual data can be used to provide additional details about what the image contains. For example, in Figure 1, we could provide additional information about the height of the mountains, their geography, or ecology. As another example, when popular individuals are named in the alt-text, their voices can be added as a voice-over to the rich audio description provided by the IMAGE Chrome extension. This allows users to get a more holistic experience that goes beyond merely describing the image. The same can be done with historical events, for example, adding a voice-over of a speech by Martin Luther King with a description of the image of it. Broadly speaking, the aforesaid tasks can be accomplished by using a combination of NLP methods to find keywords in the text and other AI methods and APIs to obtain more information on them.

Figure 3: Example of an image from Wikipedia with the alt-text "Michelle Obama and her oldest daughter walk a small black dog whilst Barack Obama and their youngest daughter walk a short distance behind them." [5]

Figure 4: Example of an image from a highly accessed blog, with the alt-text "Maine's Top Pet Friendly Attraction: Acadia National Park." [6]

2.3 Alt-text Preprocessor

This project is focused on creating a new preprocessor geared toward static images - as opposed to maps and interactive charts, which don't include alt-text. The preprocessor functions as a program running on the server. It will receive data from the client, which includes graphical information, alt-text, and some metadata. The program then attempts to store the alt-text and the image. The preprocessor continues to evaluate the compatibility of the alt-text with the image; this is to quantify how relevant the alt-text is to the image. This is done by assigning a CLIPScore to the alt-text [7]. The program proceeds to evaluate the content of the alt-text by applying Stanford Named Entity Recognition (NER) models. This is to label the named entities in the alt-text [8]. The preprocessor and implementation are further discussed in Section 3.

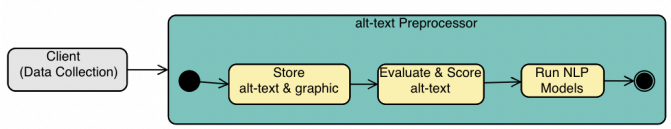

Figure 5: A high-level diagram of the alt-text preprocessor.

3 IMPLEMENTATION

3.1 External Resources

3.1.1 NER Model

Named Entity Recognition is a subtask of information extraction - the task of automatically extracting structured information from unstructured texts. Named Entity Recognition seeks to locate and classify named entities in texts into predefined categories such as person names, organizations, and locations.

In our project, we use Stanford CoreNLP - The Stanford NLP Group's natural language processing toolkit written in Java. The Stanford CoreNLP toolkit enables users to derive linguistic annotations for text, including token and sentence boundaries, parts of speech, named entities, and numeric and time values. and supports eight languages [8]. In our experiments, we use an English-named entity recognizer classifying into the (PERSON, ORGANIZATION, LOCATION, and MISCELlANEOUS) four classes; however, there are other models available for different languages and circumstances.

3.1.2 CLIPScore

In order to quantify the relevance of the alt-text to the graphics, we use CLIPScore - A Reference-free Evaluation Metric for Image Captioning [7]. Image captioning has conventionally relied on reference-based evaluations, where machine captions are compared against captions written by humans. CLIPScore, on the other hand, is reference-free, in which humans assess caption quality. Hessel et al. demonstrate that CLIPScore achieves the highest correlation with human judgments, outperforming existing reference-based metrics like CIDEr and SPICE based on experiments spanning several corpora. The CLIPScore project is accompanied by an open-source code base in Python 3. Given a path to a directory containing images and a path to a JSON file containing the captions for each image, the *clipscore.py* module evaluates the relevance of the captions and produces a score (in percentage) for each given image. The code also optionally provides other traditional caption evaluation metrics, including BLEU, METEOR, CIDER, and ROUGE [7].

3.2 Development and Integration

3.2.1 Experimental Code

The implementation of the project was done in two phases.
In the initial phase, we were focused on achieving two primary goals:

1. Writing a Python script that would achieve the main tasks of the preprocessor, namely extracting alt-text and graphics from input data, calculating CLIPScore,

and labeling the alt-text named entities using the CoreNLP toolkit
2. Curating a dataset of request files from the IMAGE Chrome extension, similar to the POST requests that will be sent to the preprocessors to test the script and calculate the CLIPScores on.

The code for the initial phase is available at https://github.com/namdar-nejad/IMAGE-context, which includes instructions on how the code can be used.

● Data Collection

In order to test our preprocessor and benchmark CLIPScores, we initially needed to collect a dataset. The data was collected using the IMAGE Chrome extension in the form of JSON files and similar to the input data that gets passed to the server preprocessors by the client.

We manually collected 80 requests from three different sources: Wikipedia, News websites, and a random selection of highly viewed personal blogs. We used these categories as they are among the most accessed web resources by the general public. Wikipedia, according to their own statistics, is the 5th most accessed website globally, with 18 billion page views per month [9]. Moreover, 63% of adults in the United States reported getting their news from a news website [10]. The Wikipedia data was collected from Fully Protected pages-pages that administrators can only edit [11]; since they have a higher likelihood of containing accuracy and unchanged alt-text. Moreover, the blog data was randomly selected from a set of highly viewed blogs [12].

● Graphics and Alt-text Extraction

As mentioned previously, the initial task of any preprocessor is to extract the necessary information from the raw data provided by the client side. In the case of the alt-text preprocessor, we need to extract the alt-text and graphics (image) from the input; they are both required to compute the CLIPScore. The *run.py* script takes in as console parameters the path to a directory containing a set of requests. The script will save the images in a .png format and captions in JSON. And the stored images and captions can be passed on to the CLIPScore code to compute the score. The essential components of the script have been reused in the preprocessor created in the IMAGE server.

● CLIPScore Analysis

The *analysis.py* script can be used to gain more information on the distribution of the CLIPScores generated. The script takes in the path to the generated CLIPScores generated at the previous step. The script retrieves the count, minimum value, maximum value, standard deviation, mean, and (25%, 50%, and 75%) percentile values.

● Stanford NER Files

This directory contains the *ner-model-english.ser.gz* and *ner-tagger.jar*, they are respectively, the NER model trained on an English corpus and the NER tagger engine written in java. The ner-tagger is called from the python script to label the named entities of the alt-text. Other models in English or in other available languages may be used.

● Notebooks

We additionally have two Jupyter notebooks, *extarct_data.ipynb* and *ner-model.ipynb*. The notebooks are playgrounds for the extraction and NER scripts and may be used to experiment with and add additional features.

3.2.2 Creating the Server Preprocessor

The next step in the implementation was to create a preprocessor (from the code written in the previous step) that could be integrated into the IMAGE server, the IMAGE project server-side code base.

As mentioned before, a preprocessor is a microservices on the server that receives HTTP POST requests containing information from the client, similar to the raw data manually collected from the IMAGE Chrome extension. The preprocessor then extracts the necessary information from the POST request and performs a specific task on them; in our case, the task is calculating a Clipscore and extracting NER tags from the alt text. If successful, it replies with a key indicating the kind of data returned and the actual content as a JSON object. This allows other preprocessors to use previous preprocessors' results to generate their own results or choose whether or not they should run their own process. The complete code can be found at https://github.com/Shared-Reality-Lab/IMAGE-server/tree/main/preprocessors/ner.

In order to create a functional preprocessor, we need a few components:

1. A Python program that is essentially the core of the preprocessor. The script receives the request from the client, extracts the data, computes CLIPScore, labels named entities, and produces an appropriate response.
2. Containerization of the preprocessor; this is done using Docker in the IMAGE server.
3. Additional scripts and files that are needed to perform the internal processing steps. This includes the code for CLIPScore and NER calculations for this project.

The initial component is satisfied by the *ner.py* script. The *ner.py* script is the main entry point of the preprocessor, where the request from the client side is received and validated. The script continues to extract the alt-text and image from the data and save them in a temporary directory. The *clipscore.py*, a simplified version of the CLIPScore project, is called, and the CLIPScore is calculated. In addition, *ner.py* includes the *stanford_ner()* that, when passed a string, will return a list of

tuples, including the word and the corresponding named entity label recognized by the Stanford model. The function is called to collect the named entities in the alt-text. Finally, the CLIPScore, NER labels, and alt-text are combined into a JSON and returned.

In addition to *ner.py, clipscore.py* has been written by simplifying the CLIPScore code base in order to reduce the complexity and size of the code. This script performs as the original project code; namely, given a path to the directory holding the images and a path, the script calculates and returns the CLIPScore. Moreover, the *Dockerfile* and *requirments.txt* are used for the automated build of the preprocessor.

The output of the preprocessor is in a JSON format that includes the keys “data”, “timestamp”, “name”, “request_uuid”. The “data” key contains an object that includes the alt-text of the image as a string, the CLIPScore as a float, and the tagged named entities. The tagged named entities are stored as an array of objects; each object includes the word that was tagged to be a named entity, the category of the named entity, and the index of the sentence in the alt-text. The “timestamp”, “name”, “request_uuid” are metadata that are used in the rest of the IMAGE pipeline. Figure 6 is the output of the preprocessor used on the original Wikipedia image shown in Figure 3.


```
{
 "data": {
  "alttxt": "\"Michelle Obama and her oldest daughter walk a small black dog whilst Barack Obama and their youngest daughter walk a short distance behind them\"",
  "clipscore": 0.941,
  "ner": [
   {
    "index": 1,
    "tag": "PERSON",
    "value": "Michelle"
   },
   {
    "index": 2,
    "tag": "PERSON",
    "value": "Obama"
   },
   {
    "index": 13,
    "tag": "PERSON",
    "value": "Barack"
   },
   {
    "Index": 14,
    "tag": "PERSON",
    "value": "Obama"
   }
  ]
 },
 "name": "ca.mcgill.a11y.image.preprocessor.ner",
 "request_uuid": "871aeb3a-3270-48b7-9a69-142fa3b71041",
 "timestamp": 1668573726
}
```


Figure 6: Example of a response from the alt-text preprocessor used on a Wikipedia image [5]

4 DISCUSSION

As mentioned in section 3.2.1, we manually collected a dataset of 80 requests from news, Wikipedia, and blog resources. We then evaluated the CLIPScore for the dataset and used the *analysis.py* script to obtain the distribution of the scores, as seen in Figure 7. We used the results to set a threshold for the CLIPScore obtained for each section.

As seen in Figure 6, from the 80 requests collected in each category, slightly over half of the requests contained alt-text, with about 60% of news and blogs and 47% of Wikipedia images containing alt-text. Moreover, the mean score is 65% for both news and blogs and 71% for Wikipedia images. The images obtained from news resources have a lower standard deviation relative to the other two resources. It can be seen that compared to news resources and blogs, fewer Wikipedia images have alt-texts, but they tend to be more descriptive and correlated to the graphics. Given the results obtained, we set a threshold of 65%, meaning that alt-texts with a CLIPScore of more than 65% are considered accurate enough to be used by the preprocessor.

Additionally, a subset of the collected requests was used to test the integrated preprocessor. The requests were forwarded to the preprocessors, and the return values were manually verified.

	news	**wiki**	**blogs**
count	46	37	45
mean	0.65	0.71	0.65
sd	0.08	0.13	0.11
min	0.53	0.47	0.44
25%	0.60	0.60	0.56
50%	0.65	0.70	0.65
75%	0.73	0.81	0.72
max	0.86	0.98	0.88

Figure 7: The distribution of the CLIPScore produced *analysis.py*

5 FUTURE WORK

The alt-text preprocessor is only an initial step in evaluating the potential of using contextual data in the IMAGE project. Although we’ve been able to integrate the alt-text preprocessor into the IMAGE server, there’s much to be done to use the results obtained from the preprocessor to improve the end product - the description provided for the graphics.

Apart from alt-text other contextual data needs to be considered, for example, the entire text of articles, the title of the page, and image captions. Moreover, the use of other NLP models and tokenizers will be necessary to extract valuable information from the texts in multiple languages. Most importantly, significant work has to be done on incorporating the results obtained from the contextual data preprocessors at the handler level.

That being said, at this stage, the contribution of alt-text to creating the outputted descriptions is more opaque than is ideal. A great subsequent step would be integrating the current processor with the handlers to output the labeled named entities. This will create the full pipeline of contextual data that can be built on and expanded.

6 CONCLUSION

We introduced the Internet Multimodal Access to Graphical Exploration (IMAGE) project and its aim to make web graphics more accessible. Moreover, we investigated how contextual data could be used to improve the descriptions produced by the IMAGE project and presented a preprocessor that works with alt-text. We found that more than half of the images found on blogs, news resources, and Wikipedia include an alt-text with a high correlation to the image graphics. The results from the alt-text preprocessor show that Named Entity Recognizer (NER) methods can extract valuable information from the alt-texts. However, at this stage, even with the evident potential of contextual data, how they can be used to produce viable end results in the IMAGE project remains to be explored. As the project progresses, we hope the ideas and preprocessor mentioned here can be used to improve the quality of descriptions produced by the IMAGE project.

ACKNOWLEDGMENT

I would like to thank Juliette Regimbal, Rohan Akut, and Prof. Jeremy R. Cooperstock for their help throughout the project.

REFERENCES

[1] Juliette Regimbal, Jeffrey R. Blum, and Jeremy R. Cooperstock. 2022. IMAGE: a deployment framework for creating multimodal experiences of web graphics. In Proceedings of the 19th International Web for All Conference (W4A '22). Association for Computing Machinery, New York, NY, USA, Article 12, 1–5. https://doi.org/10.1145/3493612.3520460

[2] Alps. (n.d.). Wikipedia. Retrieved December 19, 2022, from https://en.wikipedia.org/wiki/Alps#/media/File:M%C3%A4nnlichen.jpg

[3] Cynthia L. Bennett, Cole Gleason, Morgan Klaus Scheuerman, Jeffrey P. Bigham, Anhong Guo, and Alexandra To. 2021. "It's Complicated": Negotiating Accessibility and (Mis)Representation in Image Descriptions of Race, Gender, and Disability. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21). Association

[4] Darren Guinness, Edward Cutrell, and Meredith Ringel Morris. 2018. Caption Crawler: Enabling Reusable Alternative Text Descriptions using Reverse Image Search. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18). Association for Computing Machinery, New York, NY, USA, Paper 518, 1–11. https://doi.org/10.1145/3173574.3174092

[5] Kennedy, Chuck. "File:Obama family walks with First Dog Bo 4-14-09.jpg." Wikipedia, https://en.wikipedia.org/wiki/File:Obama_family_walks_with_First_Dog_Bo_4-14-09.jpg. Accessed 19 December 2022.

[3] for Computing Machinery, New York, NY, USA, Article 375, 1–19. https://doi.org/10.1145/3411764.3445498

[6] "Pet Friendly Campgrounds At America's National Parks." GoPetFriendly, 15 August 2022, https://www.gopetfriendly.com/blog/pet-friendly-campgrounds-americas-national-parks/. Accessed 19 December 2022.

[7] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. 2021. CLIPScore: A Reference-free Evaluation Metric for Image Captioning. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 7514–7528, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

[8] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

[9] West, A. (n.d.). Wikipedia:Statistics. Wikipedia. Retrieved December 20, 2022, from https://en.wikipedia.org/wiki/Wikipedia:Statistics

[10] News Platform Fact Sheet. (2022, September 20). Pew Research Center. Retrieved December 20, 2022, from https://www.pewresearch.org/journalism/fact-sheet/news-platform-fact-sheet/

[11] Wikipedia:Protection policy. (n.d.). Wikipedia. Retrieved December 20, 2022, from https://en.wikipedia.org/wiki/Wikipedia:Protection_policy#full

[12] (n.d.). Discover Awesome Bloggers - Top Blogs of 2021 - RankedBlogs.com. Retrieved December 20, 2022, from https://www.rankedblogs.com