TEAM 1 of KAIST CS 492(I) : Intro to Deep Learning
Research Track - Final Report

Is ALT Text Really an Option? A Novel VScode Extension for Context Aware Image Captioning

Inhwa Song(20190325)*, **Jiho Park(20210264)***, **Injoon Hwang(20210705)***

*Undergrad in School of Computing, KAIST

1. Abstract

Alt text is an interface for visually impaired people to get a sense of web image. Unfortunately, majority web content generators lack motivation to endure the burden of inserting alt text, and external alt text generator tools on the web lack quality, especially the context information, so human in the loop is highly required in the process. We present *Altelper*, a VScode extension that enables HTML script producers to insert contextual alt text attribute in img tag with dramatically less burden. Also, we implement context aware image captioning model from scratch, referring to the structure introduced in Chowdhury et al.[1], as thereby eventually accomplishing not only requirements of Research Track, but part of Implementation Track as well. By interpreting the users' goal with an image captioning model embedded VScode extension, *Altelper* supports image caption recommendation that the user can apply in their context aware image captioning. A comparative study(N=18) showed that *Altelper* supports participants to perform context aware image captioning faster and richer than writing from scratch.

2. Introduction

It is undeniable that the web is now a critical part to every-day tasks. A novel characteristic of the web is that while large entities - The Government, Schools, Google, Facebook, Amazon etc. handle most of the traffic, small groups or even individuals are able to create services that have a large impact especially on small communities. While serving as a great source of diversity & innovation, this means that maintaining good accessibility throughout the entire web is very difficult. We conducted a survey of 31 developers where only 5 took accessibility as a consideration in their projects. Also, only 18 out of 100 randomly sampled websites met the level A, which is an official web accessibility standard. A disturbing trend was also found with code linters and automated IDE tools. Modern linter rules and IDEs try to force compliance to accessibility guidelines using warning messages and automated code skeleton generation while a significant number of developers put nonsensical text (e.g. alt="img") to fulfill the requirements. While seemingly harmless, this worsens accessibility by mitigating automatic inference tools provided by browsers and screen readers.

Through individual interviews, we were able to identify two key factors of this problem. First, we must minimize the barrier in writing proper alt texts. Second, we must provide an incentive for developers for writing proper alt texts. Recent research have proposed methods of implementing these solutions in the browser or screen readers. While these solutions have their merits, it passes the burden of solving the problem to the consumer. Also, it still does not solve the problem of improper use of alt texts.

We propose implementing the solution as a form of a VSCode extension. VSCode is the most popular web IDE and has a large user base of people starting to learn web development. By implementing the solution directly in the development tool, we can 1) ensure that all consumer mediums properly support accessibility features and 2) increase awareness about web accessibility.

3. Methods

3.1. Altelper

Altelper(previously named ICS in the previous presentation), is a novel VSCode extension that helps developers automatically write good Alt Text within their HTML code. While many A.I. based solutions are already available for browsers and screen readers, *Altelper* is the first solution that directly integrates with the IDE as far as we know.

In order to prove that our concept had merit, we first

developed the *Altelper v1* prototype using a pretrained image captioning model from huggingface. After trials, we were excited to see that the prototype meaningfully helped developers write faster, context aware, and quality alt texts and the reception was generally positive. We then identified areas we could improve on and are working on *Altelper v2*. This version supports context-aware image captioning using a custom model based on Chowdhury et al.[1] and can automatically collect dataset from 1) user choice on recommended options and 2) code snippets. This dataset can be used to further improve our generation models. Unfortunately, there was an issue with integrating the newly trained model with the backend system. As we're motivated to release this project to the public, we're working hard to resolve this final issue.

3.2. Pipeline

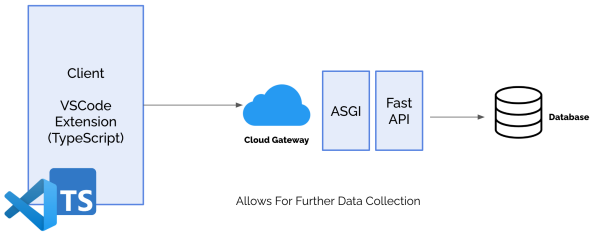

Figure 1. v1 architecture of *Altelper*

3.2.1. Overall Architecture

We implemented the interface of *Altelper* as a VScode extension, using typescript and VScode SDK. To serve the model pipeline, we use a server running FastAPI with Uvicorn ASGI interface and deployed 1) RabbitMQ with Pytorch to run the model 2) MongoDB to log user choice.

This pipeline allows us to develop a richer dataset and allows for greater flexibility in serving the model. As we plan to deploy *Altelper* to VSCode marketplace, if sufficient userbase is formed, we can easily transition to distributed computing systems or even serveless models.

3.2.2. Client Side

Altelper detects img tag and former div tag of the image. It parses each src attribute(image web url) and content inside the div tag, and passes it to the server using Axios API. It is compatible not only to the pretrained vit-gpt2-image-captioning model where only src input is required, but also scalable to our own developed model where both src attribute and context paragraph is required as input. Once the client receives the alt text recommendation result, it provides the user via VScode command palette, and allows the user to modify and apply it to their code with a single click.

3.2.3. Server Side & MLOps

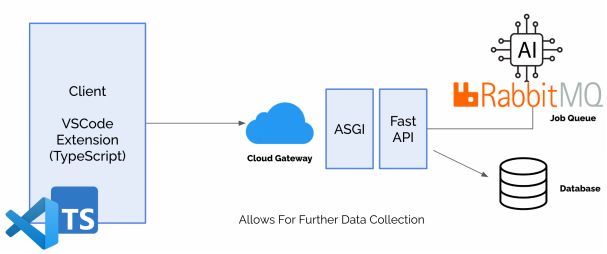

Figure 2. v2 architecture of *Altelper*

In version 1, FastAPI exposes a REST endpoint *predict* through uvicorn that takes an image URL as argument. FastAPI then directly runs the pretrained model using *pipeline* by huggingface. The downside to this approach is that 1) the system (the endpoint) is bottle-necked by running the model and 2) deployment becomes difficult as the entire server must also support CUDA drivers. (Note Figure 1.)

In version 2 (In development), we introduce RabbitMQ and Docker Swarms to 1) support data collection and 2) solve the problems in version 1. FastAPI now dispatches a job to RabbitMQ (Job Queue) using *pika*. The job is dispatched to one of the workers in the Docker Swarm using the Round-Robin algorithm. The status is reported back to the client asynchronously - we provide endpoints to poll the server about the status of the job. This allows for a better user experience as the user does not have to wait for the server to respond and also allow for a higher throughput as more workers can be dynamically added to the swarm as needed. We also provide endpoints for data logging that store data collected from users in MongoDB for further model improvements. (Note Figure 2.)

3.3. Context Aware Image Captioning

3.3.1. Background

For rich captioning support, we tried to implement a model that understands the context of an image. Referring to Chowdhury et al.[1], we found its repository. However, there were no instructions about the usage and list of required packages. Moreover, it did not run without several code and configuration modifications. Dataset used posts of the "r/pics" subreddit, but there

was no image crawling and feature extracting codes. Therefore, we needed to implement the paper's model from scratch.

Implementing the model based on the paper and the code of its repository, we found that they do not match. For instance, the calculation of attention was different. Thus we adopted the basic architecture and tried to implement it in our way.

3.3.2. Implementation

The first thing we did was crawl images from Reddit and extract their features using ResNet-152. As the training and test datasets were valid, we first tried to crawl them. However, as the size of the training set was 30,000, memory ran out, and the process was killed after processing about 1,000 images. Therefore, we used a small dataset with 160 image URLs provided by the original research.

Around this time, we emailed the researcher of the original paper, if we could get any pre-trained model. She replied that there were none and that the codes on the repository were all she had. After this mail, we forked the repository and focused on modifying the codes to make the code runnable. Fortunately, it worked, and we used the model for *Altelper v2*.

4. Evaluation

We conducted a between-subjects study where we compared *Altelper* against writing alt text from scratch. The study was composed of 2 well-defined tasks of generating alt text of a given context and image. We designed these tasks to investigate how *Altelper* helped participants to (1) enhance overall speed of image captioning considering not solely the image but the overall context, (2) help with higher quality image captioning in a limited amount of time, and (3) affect self-confidence of their image captioning abilities. Overall, we referred to the evaluation method applied in Kasai et al.[3], Hart et al.[2], Kim et al.[4], Bertram et al.[5]

- RQ1. Can *Altelper*, that gives only image based caption recommendation help users produce context aware caption with high speed with reasonable quality?
- RQ2. Can *Altelper*, that gives only image based caption recommendation help users produce caption with higher quality than writing from scratch?
- RQ3. How does usage of *Altelper* affect their self-confidence regarding their captioning abilities?

4.1. Participants

We recruited 18 participants(6 female, 12 male) who all reported having experience with VScode and HTML. Participants were divided into two equally-sized groups and each group was assigned to use either *Altelper*(Group A) or no support tool(Group B). In advance of the main task, we conducted a brief walk-through workshop for both Group A & B who are novice to *Altelper* to get used to the tool.

4.2. Study Procedure

The study took place face-to-face for 8 participants, and remotely sharing screens for 10 participants. Each participant was guided to run *Altelper* on their VScode in advance. Participants were first provided with a brief walkthrough of their assigned tool and were allowed to test the tool for a total of 5 minutes. After this, participants completed a short pre-task survey.

After the survey, participants started Task 1. Participants were tasked with using either *Altelper*(experimental group, 9 people) or no tool(control group, 9 people), given enough pairs of (image, text) to reproduce via HTML code. The (image, text) pairs were randomly selected from the dataset presented in Chowdhury et al.[1] where they scrolled from Reddit. They were given to reproduce as many (image, text) pairs as possible considering context and image in 3 minutes.

After Task 1, participants started Task 2 after a 3-minute break, where they were tasked to reproduce 3 (image, text) pairs within 2 minutes. Finally, a short interview was conducted asking participants about their experiences during both tasks. Also, a post-survey was conducted after the 2 tasks.

4.3. Measures

We collected responses to the pre-survey and the post-surveys after each task. All of the surveys contained 3 questions asking participants to rate their self-confidence with respect to their ability to (1) produce context aware alt text, (2) produce image representative alt text, and (3) produce reasonable quality of alt text overall on a 7-point Likert scale. We averaged the responses to these questions to derive one score for self-confidence. The post-survey included the six questions from the NASA-TLX questionnaire from Hart et

al.[2] to measure perceived workload of the participants.

We also quantitatively measured task-related metrics. For Task 1, we measured the number of (image, text) pairs reproduced in a given amount of time. We hypothesized that *Altelper* would help participants produce context aware alt text faster, given an initial version containing image based caption, and therefore complete the production in less time.

For Task 2, we measured the quality of alt text by scoring of crowds(18 that we recruited and our own) by the survey. The quality measurement survey included questions from THUMB, a rubric-based human evaluation protocol for image captioning models from Kasai et al.[3]. Since we also measure context awareness of the result, we added an evaluation entry for context awareness.

Though 21 people are not an enough number for trustworthy quantitative evaluation, not given enough financial resources to recruit people nor use general crowdsourcing platform(e.g. MTurk), 21 was our maximum effort.

For qualitative data, we analyzed participants' responses during the short interviews to understand their perceptions of the given tool and how they leveraged it for their purposes.

5. Result & Analysis

Our result demonstrated that *Altelper* helped participants perform context aware image captioning faster and with greater quality in Task1 and Task 2.

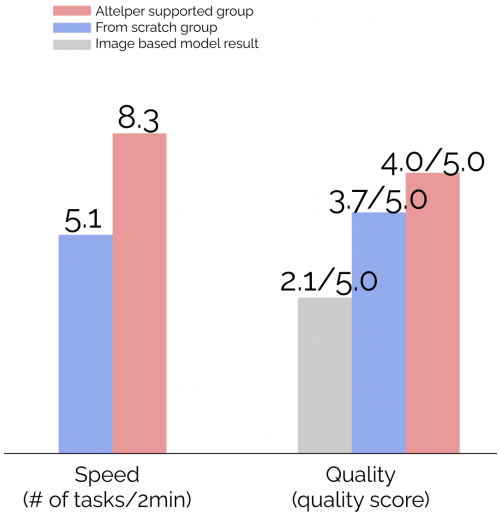

Figure 3. Comparative study result

5.1. Speed and Quality

To answer RQ1, we analyzed participants' speed in Task 1.

Overall, *Altelper* participants(Group A) significantly outperformed from scratch participants(Group B) in this task. While the average number of (image, text) reproduced in 3 minutes was 5.1 for Group B, but 8.3 for Group A(Figure 3).

To answer RQ2, we analyzed participants' caption quality in Task 2. We compared three groups of captions. Raw captions generated by image based model, produced from Group B, and produced from Group A. Group A outperformed both in means of quality, and the order of quality is as Figure 3.

5.2. Self Confidence

To answer RQ3, we evaluated how self-confidence changed during the study by analyzing differences in participants' responses. Group A(*Altelper*) participants' self-confidence increased significantly between the pre-survey, and the post-survey. Participants felt satisfied about completing the tasks, and mentioned that it was easy to produce alt text using *Altelper*: "I wouldn't be bothered inserting alt text with *Altelper*, but not sure without it." (S3). Participants' self-confidence who produced alt text from scratch slightly decreased between the pre-survey and the post survey. Furthermore, the perceived workload of context aware image captioning showed a significant gap between two groups, Group A's average perceived workload smaller than Group B.

6. Limitations

Our work has several limitations which we address in this section. With no financial support, we couldn't use crowdsourcing platforms(e.g. MTurk) to recruit crowd workers. Instead, we recruited our friends and people nearby for formative study, evaluation, and caption quality measurement tasks. Therefore, our core limitation would lay beyond the fact that not enough compensation was provided for recruited participants which has the possibility to lower the task quality, and also raise bias of the participant pool.

7. Contributions & Github Repositories

7.1. Inhwa

- VScode extension implementation by typescript

- Communicated with the paper[1] author via email regarding incompatible code of the open sourced github repo
- Overall task management
- Evaluation design
- Evaluation conduction

7.2. Jiho

- Problem recognition & research on ALT text, web accessibility
- System design - Divided project into Extension, Backend Model
- Backend, MLOps implementation (Integrated a working demo of the entire project)
- Evaluation conduction

7.3. Injoon

- Implementation of contextual image captioning model
- Made the code of the original paper runnable, which had not run without any code modifications
- Evaluation conduction

7.4. Github Repository

- Client Side: https://github.com/greenina/Altelper
- Preprocessed Data Generation: https://github.com/injoonH/contextual-image-captioning
- Context Aware Image Captioning Model: https://github.com/injoonH/contextual_captions
- Server Side & MLOps: https://github.com/UrWrstNightmare/cs492I-project-backend

8. Acknowledgements

We highly thank and respect Professor Seunghoon Hong and the TAs for coordinating such a gainful course.

References

[1] Chowdhury, S. N., Bhowmik, R., Ravi, H., de Melo, G., Razniewski, S., & Weikum, G. (2021, April). Exploiting Image–Text Synergy for Contextual Image Captioning. In Proceedings of the Third Workshop on Beyond Vision and LANguage: inTEgrating Real-world kNowledge (LANTERN) (pp. 30-37).

[2] Hart, S. G., & Staveland, L. E. (1988). Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In Advances in psychology (Vol. 52, pp. 139-183). North-Holland.

[3] Kasai, J., Sakaguchi, K., Dunagan, L., Morrison, J., Bras, R. L., Choi, Y., & Smith, N. A. (2021). Transparent human evaluation for image captioning. arXiv preprint arXiv:2111.08940.

[4] Kim, T. S., Choi, D., Choi, Y., & Kim, J. (2022, April). Stylette: Styling the Web with Natural Language. In CHI Conference on Human Factors in Computing Systems (pp. 1-17).

[5] Bertram, D. (2007). Likert scales. Retrieved November, 2(10), 1-10.