Validating UI requirements using visually-grounded language models

Carlo Angel Lujan Garcia, Ricardo Gonzalez Leal, Luis Alberto Alcántara Cabrales, Karla Gonzalez Sanchez, Isai Ambrocio, Luis David Lopez Magaña

Abstract

Verifying that a web page meets the visual requirements is an arduous task that usually takes a long time, due to the fact that they are performed manually by humans. That is why this study addresses this problem by means of automatic evaluation of requirements, without the need to interact directly with the DOM of the website. For this reason, we have proposed a Google Chrome extension, which takes a screenshot of the website the user is visiting and allows to enter the requirements through an interface or by uploading a text file. The extension uses the LlaVA model with fine-tuning to analyze the image and determine whether the requirements have been met or not. Its contribution to the testing field allows for greater efficiency by not relying on the DOM, as well as its easy maintenance and focus on the non-functional visual part.

Isai Ambrocio
ITESM, Av. Gral Ramón Corona No 2514, Colonia Nuevo México, 45201 Zapopan, Jal., e-mail: A01625101@exatec.tec.mx

Luis Alberto Alcántara Cabrales
ITESM, Av. Gral Ramón Corona No 2514, Colonia Nuevo México, 45201 Zapopan, Jal., e-mail: A01634185@exatec.tec.mx

1 Introduction

Software testing has a fundamental role in ensuring the quality and reliability of a system or software, with the objective of guaranteeing its correct operation and avoiding causing interruptions that can generate many costs. In addition, according to Serna et al. (2019)[1], many authors and researchers agree that software testing is fundamental, as it helps to detect defects, reduce associated risks, is the key to the success of the development of systems, so that the customer is satisfied and helps to ensure quality which has a direct impact on the cost and development of the product. And, according to Menejías García et al. (2021) [2] , as quality testing is the main way to detect errors and vulnerabilities, the software industry recognizes its importance.

Nowadays there are several ways and means to test software, such as testing software based on visual information, which allows to evaluate the interface and user experience in a much more effective way, can help to identify usability, accessibility and design problems that could directly or indirectly affect the satisfaction of end users. And, according to Fernandez (2023) [3], visual testing helps to avoid generating thousands of lines of code when testing a system such as a web page with many elements of which we want to analyze characteristics such as visibility, height, width and background color, and also helps to avoid having to make many changes in the test code every time a new version of the system is generated, avoiding falling into a situation in which the maintenance of the tests is unfeasible.

And, although there are several ways to do software testing, currently in general there are many challenges in relation to software testing, since, as mentioned by Mamani (2023) [4], after the pandemic of 2020 there was a big change in the management of information generating a greater demand for users in applications which caused the need for greater efficiency and maintainability of application services, Therefore, the use of microservices was adopted, which have many challenges, such as software testing, since many apps have thousands of services, which would require investing a lot of time in testing and could become almost impossible. In addition, according to Menejías García et al. (2021) [2], other challenges that currently exist in relation to software testing are to reduce and even eradicate the practice of performing software testing at the end of the software development cycle and only perform functional tests.

Also, according to Patel (2024) [5], with continuous development and the high frequency of updates, keeping test cases up to date and performing software testing becomes very complex and very costly (mainly in regression testing). And, according to Zap Chernyak (2024) [6], performing manual UI or design testing is time-consuming, prone to human error, some bugs may be missed due to inattention or distraction, and a lot of human resources are needed. Taking up the above, with so many challenges, it can be said that it is necessary to look for a new approach to software testing and always ensure quality, even as the number of systems and programs continues to grow.

The approaches that have been implemented to solve these problems have been to automate software testing, as mentioned in the article by Serna et al. (2019) [1], which has brought good results and benefits, but it has its limitations such as it cannot replace manual testing, the expected objectives cannot be achieved because by executing tests in a very short time it is not possible to obtain lasting or real benefits, they are very difficult to maintain due to the various changes in technology and the great evolution of software, and it is very difficult to automate visual acceptance, interface and UI tests with traditional programming methods.

With the new model generated from the pre-trained multimodal model LLaVA, we seek to overcome the limitations of traditional automated testing with the use of a visual artificial intelligence that automates and optimizes the time of the software testing process. This can be obtained, since a robust model such as LLaVA, if well trained, can become a very complete option for visual testing, as it could allow the accurate identification of errors and discrepancies in the user interface in a short period of time. Such a model will not only help to streamline the testing process, but will also reduce human error, thus ensuring that software products are of higher quality and are more reliable.

First, in the ”Related Works” section, previous works on the topic of visual software are analyzed, comparing computer vision approaches and methods, to give context to the research. Then, in the ”Methodology” section, we detail the steps and techniques used to collect data, the selection of computer vision tools and natural language processing (multimodal model), and additionally include the design of experiments to evaluate the effectiveness of our approach in visual software testing. Then, in the ”Implementation” section, we detail how the multimodal model was generated from the pre-trained LLaVA model to create a visual software testing system including configurations and other related information.

Then, in the ”Experiments” section, you can find the details of the design, datasets, comparison with other methods and metrics to validate the implemented multimodal model. Then, in the ”Results” section, the experimental results are presented, analyzing the performance of the model with some metrics and illustrating its effectiveness with use cases, as well as using visual supports such as tables and graphs.

In addition, the ”Discussion” section details the discussions regarding the implications of the results in the software testing landscape, analyzing the strengths and weaknesses of the proposed model and suggesting directions for future research with its limitations. And finally, in the ”Conclusion and Future Work” section, the key conclusions of the study are synthesized, focusing on the potential impact of the developed model on the software testing industry and establishing future research directions.

2 Related Works

For the visual tests that correspond to the GUI,[7] usually a graphical interface and an expected mockup is used, which is usually integrated in some automated testing frameworks such as Cypress, Selenium Web-Driver, among others.[8] There are also the traditional methods to check that the CCS corresponds to the HTML page element.[9, 10]

On the other hand, the approach of our project is different because only a screenshot of the web page is taken, which is being browsed at the moment and by means of an input of requirements through the graphical interface of the extension or by means of a txt file. The model, being multimodal,[11] analyzes the screenshot with the requirements provided, which allows us not to depend on other frameworks or provide any mockup but only text in natural language.[12]

The mathematical part where the loss function calculations of the Large language models come in was very interesting, as it indicates the sum of the negative log probabilities of the next token.[13]

It has been seen in other articles that LLaVA has different versions, some with a great performance in multimodal task dialogs, there are even those who have managed to make it potentially smaller than other models with only 2.7B of parameters.[11] This allows greater agility dialogs with textual and visual elements.[14] Also, being open source, the community makes its adaptations to the original model, which allows having a wide range of options and combinations between models for different approaches[11] and specializations in certain areas.[9, 14]

One of the findings we saw is that it is not entirely necessary to use the DOM to be able to perform visual testing of elements, and it can be faster to analyze using a screenshot.

However, the limitations we presented was to find a suitable and wide enough dataset to be able to do the fine-tuning[14] in a more adequate way. Also, we could observe that there were models which were much more optimized by reducing the parameters,[11] something that our model had, but in a very simple way; this was observed due to the high use of dedicated GPU needed, so one of the limitations of LLaVA is its high consumption of computational resources.

3 Methodology

LLaVA is based on a transformer architecture. This model is an open source multimodal chatbot where the deep convolutional network (CNN)[15] is in charge of processing and analyzing screenshots, while the LLM[12] is in charge of understanding the context and generating more complete descriptions and explanations.

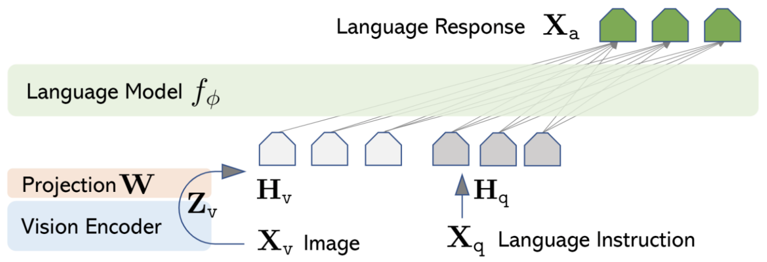

Fig. 1 LLaVA architecture [14]

An additional computational vision model[14] was also used, which is capable of associating images with textual descriptions.

That said, the techniques used were computer vision, which allows identifying and locating elements within the screenshot and then making the comparison between the expected elements and those present in the screenshot. The model selected for this task was OpenAI-CLIP.

On the other hand, with the deep learning[15] and the adaptation by means of fine-tuning done to the model, an improvement in terms of accuracy in the evaluation of specific requirements of the web interface was sought. For this purpose, the DeepSpeed library was used, whose function is to optimize the training of large models, allowing them to be more efficient and faster, using parallelism and memory optimization techniques.

Therefore, it was also necessary to process the integrated requirements in text form, using natural language processing (NLP)[14] techniques to extract and understand the specifications.

In turn, a multimodal data integration was performed, in other words, combining image and text data in order to provide a holistic assessment of compliance with the requirements.

Furthermore, as far as screenshot processing and analysis is concerned, the way it works is as follows. The extension took a screenshot of the current web page, storing it temporarily for analysis. The image is then processed to normalize its size and resolution, ensuring consistency in the analysis, and the Llava model is used to analyze the image in detail, detecting and classifying the visual elements according to the specified requirements. Subsequently, the highlighted elements are compared with the entered requirements.

In addition, a dataset with different categories was used so that the model can distinguish from a button to a text box, from a login to a navigation bar, entire websites, etc. This is so that it can indicate errors or problems in the user interface because it would not meet the requirements stipulated.

4 Implementation

Model: LLaVA 1.5 was used, which is a combination of a Large Language Model - Vicuna (based on LLaMA) - and a visual encoder - CLIP ViT-L/14. The union of the visual and textual components creates a broader context that allows generating more informed and complete responses. Additionally, the main attraction of LLaVA is its quality as open-source software, which allows fine-tuning for specific situations and complex contexts.

In this project's case, the images that the encoder will handle contain common web elements such as buttons, forms, headers, footers, and labels, among others; in addition to unique features like color and position.

To generate a comprehensive and readable dataset for the model, the UBIAI labeling tool is used, which simplifies the process of filling attributes and generates quality information. The script that serves as a template can be found at the following link: https://ubiai.tools/how-to-fine-tune-llava-on-your-custom-dataset/
Backend: Python script that uses the FastAPI library to create a simple REST API. It uses both a fine-tuned LLaVA model and GPT-3.5/4 to make descriptions of webpages and compare them with the provided requirements, respectively. It consists of four endpoints:

- GET /: API root, returns a welcome message.
- GET /screenshot/: allows uploading webpage screenshots to Cloudinary.
- GET /webElement/: similar to previous but specifically for web elements.
- GET /gpt3/: generic call to GPT3, returns a response generated by the model.
- GET /llavaft-describe/: call to the fine-tuned LLaVA instance that requests a description of the UI present in the attached image (website to be analyzed).
- GET /llava-describe/: call to the regular LLaVA model that requests a description of the UI present in the attached image (website to be analyzed).
- GET /ui/: same as previous but calls GPT4 Turbo.
- GET /compare/: call to GPT4 Turbo that requests a comparison and determines the similarities and differences between the description generated by the previous call and a user-provided description. It returns descriptions of the elements of interest and their compliance.

Frontend: Google Chrome extension that is deployed over the website of interest and allows access to the comparison model. The interaction process consists of the following steps:

1. The user accesses the extension.
2. The user inputs a series of requirements that the page should meet in the form of a description in the text area/uploads a .txt file.
3. Once entered, the user clicks the button and obtains the comparison.

The client follows the following process to obtain the result: —

1. A screenshot is taken and uploaded to the Cloudinary platform.
2. A call is made to the third endpoint with the predefined prompt referring to the newly generated image of the site.

3. Once the response is obtained, a call is made to the fourth endpoint with both descriptions: the newly obtained one and the user-provided one.
4. Finally, the result is displayed in the extension.

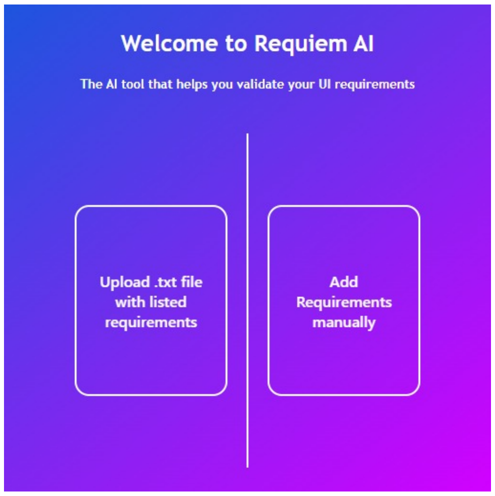
Welcome to Requiem AI
The AI tool that helps you validate your UI requirements
Upload .txt file with listed requirements
Add Requirements manually

Fig. 2 RequiemIA main interface.

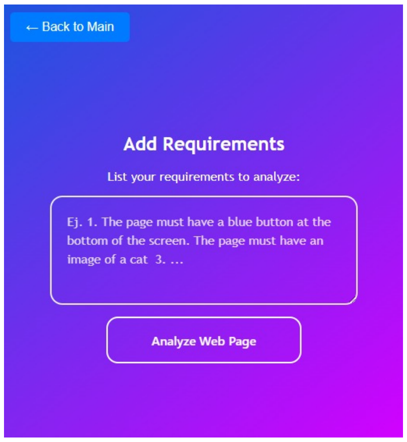
← Back to Main
Add Requirements
List your requirements to analyze:
Ej. 1. The page must have a blue button at the bottom of the screen. The page must have an image of a cat 3. ...
Analyze Web Page

Fig. 3 RequiemIA manual requirements interface.

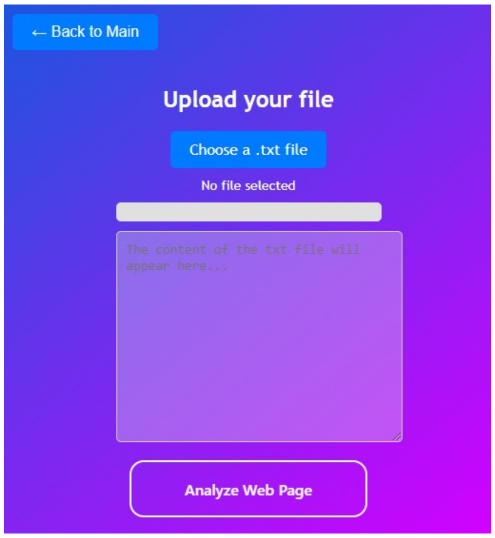

Fig. 4 RequiemIA upload text-file interface.

5 Experiments

Experiments to validate the model created were contrasted with the original LLaVA model. In these tests, different performance metrics will be measured as function loss, processing time, scalability, and accuracy.

The dataset used for the project was taken from the Roboflow platform. This dataset, which was published by the author "Test", is licensed under "CC BY 4.0"[16] which allows distribution, remixing, adapting and building, even commercially as long as credit is given to the author for the creation of the original.

Taking into account the above, it was decided to reduce the dataset in the different categories it has, this is because when performing a visual exploration the dataset presented many low resolution images. In addition, some classes such as login were intentionally reduced to improve the learning of the model, so the dataset is not balanced.

The dataset has 9 categories, which correspond to 200 images for buttons, 100 logos, 200 magnifying glasses, 200 nav-blocks, 200 options, 200 nav-links, 200 for inputs, 200 for web pages, and 200 for search fields.

Since we are working with images and not with stochastic data, there is no way to determine the level of randomness and to know if they follow a para-metric distribution to which they can be fitted, and it is not possible to determine if it is not parametric.

However, autoregressive cross-entropy was used to determine the loss function. This is a measure of the difference between two probability distributions, which are the actual distribution and the distribution predicted by the model, and is autoregressive as it generates each part of the outputs. Sequentially based on the previous parts generated. [13]

$$L[\phi] = -\sum_{i=1}^{I}\sum_{t=1}^{T} \log\left[Pr(x_{i,t+1} \mid x_{i,1\ldots t}, \phi)\right]. \tag{1}$$

For the accuracy, the correct predictions were provided, which are made up of the completed and not completed predictions divided by the total predictions.

$$Accuracy = \frac{TruePositives + TrueNegatives}{TotalPredictions} \tag{2}$$

6 Results

The results show a slight improvement to the original model; however, it was a minimal improvement in most of the items. The results of the models will be detailed below.

In addition, LoRA was used to reduce the number of parameters to train in the model, slightly improving efficiency without significantly compromising performance. This helped reduce our loss function.

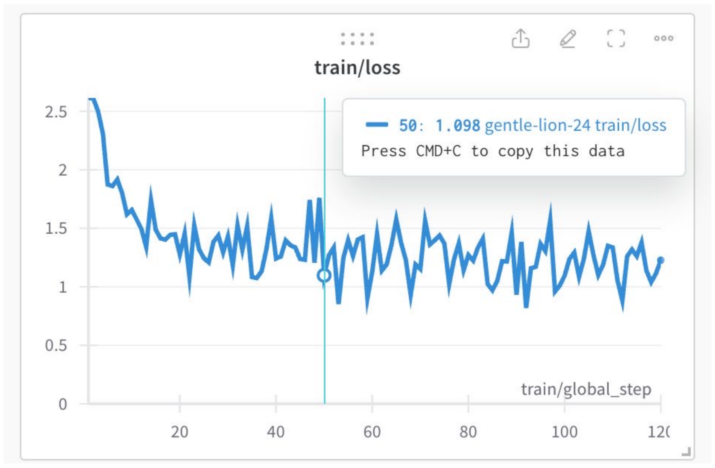

Fig. 5 Autoregressive cross-entropy graph

Also, a minimal improvement in processing speed was seen as the original model took an average of 2.01s using the gradio client and 12.24s with the API, while our model took 2.35s using the gradio client and 9.30s with the API.

As for scalability, it remained practically the same even with the reduction of some parameters in the use of GPU resources between both models, which is a great point to keep in mind and improve in future work, due to the high demand.

Aditionally, the accuracy of the original model obtained a score of 80.71% while the fine-tuned model obtained only 82.13% due to the small amount of new data provided to the model due to the small amount of datasets suitable for our approach. Even, the data set we used had to be trimmed, since it had low resolution images.

Finally, the use cases were based on 20 tests where the fine-tuning model was able to give slightly more accurate results, which means that it was better able to determine which requirements were met and which were not.

7 Discussion

The proposed model was slightly different in some items compared to the original model. However, they were not highly significant, so the original model could be used without any drawback.

Some of the advantages of our model are the slightly more accurate accuracy, a decrease in the loss function, as well as a decrease in processing time of almost 3s with the API.

The weakness of the model would be the minimal improvement in each item mentioned above, since fine-tuning did not have a major impact. Even in items such as client radio it is slightly lower by almost 0.30s.

Future work could extend the dataset synthetically through data augmentation along with new real data, which could make a considerable change in the accuracy of the model. On the other hand, trying to reduce the CPU usage and optimize the model through algorithms and parameter reduction so that it can process even faster than the current one.

8 Conclusion and Future Work

Finally, several conclusions can be drawn from this study, such as:
Key findings:

- Visual test automation with artificial intelligence has shown significant improvement in identifying discrepancies in graphical interfaces.
- Parameter optimization with techniques such as LoRA is a way to reduce the number of model parameters without significantly affecting model performance.
- The use of deep neural networks improves the accuracy of visual error detection in web applications.

Limitations encountered:

- The data set used is very limited and of not so good quality which somewhat affects the accuracy of the model.
- This type of multimodal model and deep learning requires a large amount of computational resources.
- If the project is to be done on a large scale, the processing time can be very long and become a significant constraint.

How to solve the limitations:

- Extend the data set with the highest quality data already available using methods such as data augmentation.
- Implement resource optimization techniques such as the use of LoRA to reduce model complexity and quantization techniques to optimize the use of computational resources.
- The optimization techniques that were employed have significantly reduced the processing time, but there is room for improvement and other techniques can be sought to make the processes more efficient.

The developed model can have a great positive impact on the software testing industry, as it can significantly optimize visual testing by making it automatic and with the efficiency of an artificial intelligence, which can decrease costs in the testing process and the amount of human resources for testing. In addition, our approach can optimize the acceptance testing process in web page design issues, as it can identify the most important GUI elements of a web page with their main characteristics such as color, size, shape, text, etc. This technological advance in the software and AI industry not only speeds up the development cycle of a system or software, but also improves accuracy and consistency in visual error detection, resulting in higher quality of the final product. And by automating repetitive and detailed tasks, the model frees software testers to focus on more complex and creative areas of the testing process.

As it can be concluded this type of approach to the subject of software testing, mainly visual testing, can serve to positively impact the industry so it is recommended to refine and follow the project to make it better, more accurate, more precise and more efficient. One path that can be taken is to investigate new optimization techniques that can increase model accuracy and reduce processing time, which would include the development of more advanced learning algorithms and the use of specialized hardware. In addition, outside the topic of visual software testing and visual testing for web pages, another avenue of research could be to seek to adapt and expand the use of LLaVA to other domains, such as graphic design, anomaly detection in medical images, and task automation in the manufacturing industry. And finally, another path that can be taken is to look at how to integrate the model with other AI technology to create more complete and multifunctional solutions, such as creating a system with a synergy between LLaVA and GPT 4.

References

1. E. Serna, R. Martínez, and P. Tamayo, "Una revisión a la realidad de la automatización de las pruebas del software," *Computación y Sistemas*, vol. 23, no. 1, pp. 169–183, 2019.
2. R. Menejías García, N. H. Hidalgo Reyes, A. Marín Díaz, and Y. Trujillo Casañola, "Procedimiento para evaluar seguridad a productos de software," *Revista Cubana de Ciencias Informáticas*, vol. 15, pp. 333 – 349, 00 2021. [Online]. Available: http://scielo.sld.cu/scielo.php?script=sci$_{a}rttext pid = S2227 - 18992021000500333 nrm = iso$
3. Fernández Villar, "Testing automatizado de software basado en ia," 2023. [Online]. Available: https://www.caixabanktech.com/es/blogs/testing-automatizado-de-software-basado-en-ia/
4. C. A. L. Mamani, "Pruebas de software para microservicios," *Innovación y Software*, vol. 4, no. 1, pp. 151–160, 2023.
5. R. Patel, "5 major software testing challenges in 2024," 2023, Última actualización: 3 de junio de 2024. [Online]. Available: https://www.spaceo.ca/blog/software-testing-challenges/
6. A. Zap Chernyak, "¿qué es la prueba de software de interfaz de usuario? profundización en los tipos, procesos, herramientas y aplicación," 2024. [Online]. Available: https://www.zaptest.com/es/que-es-la-prueba-de-software-de-interfaz-de-usuario-profundizacion-en-los-tipos-procesos-herramientas-y-aplicacion
7. T.-H. Chang, T. Yeh, and R. C. Miller, "Gui testing using computer vision," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 1535–1544. [Online]. Available: https://doi.org/10.1145/1753326.1753555
8. I. S. Y. K. Kateryna Ivanova, Galyna Kondratenko, "Artificial intelligence in automated system for web- interfaces visual testing," 2020.
9. T. Graves, M. Harrold, J. Kim, A. Porter, and G. Rothermel, "An empirical study of regression test selection techniques," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 10, no. 2, pp. 184–208, 2001.
10. J. Eskonen, J. Kahles, and J. Reijonen, "Automating gui testing with image-based deep reinforcement learning," pp. 160–167, 2020.
11. Y. Zhu, M. Zhu, N. Liu, Z. Ou, X. Mou, and J. Tang, "Llava-phi: Efficient multi-modal assistant with small language models," 01 2024.
12. H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," 2023.
13. S. Prince, "Training and fine-tuning large language models," https://www.borealisai.com/research-blogs/training-and-fine-tuning-large-language-models/, March 2023, borealis AI.
14. H. Liu, C. Li, Y. Li, and Y. J. Lee, "Improved baselines with visual instruction tuning," 2023.
15. L. Alzubaidi, J. Zhang, A. J. Humaidi *et al.*, "Review of deep learning: concepts, cnn architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 1, p. 53, 2021. [Online]. Available: https://doi.org/10.1186/s40537-021-00444-8
16. test, "webpages dataset," https://universe.roboflow.com/test-gktvm/webpages-xuiyu , oct 2023, visited on 2024-06-11. [Online]. Available: https://universe.roboflow.com/test-gktvm/webpages-xuiyu