Similarity Detection in Android Screen: a comparative analysis using Information Retrieval, Embeddings and Vision Language Models

Daniel Augusto R. Lima de Souza, Fabio C. Ramos, Cainã S. de Oliveira, Edluce L. Veras
Paulo Fabricio da F. Lopes, Barbara L. Santos, Jose Diogo B. de Souza, Adriano De O. Oliveira
Sidia Science and Technology Institute, Manaus, Brazil
E-mail: {daniel.souza, fabio.ramos, caina.oliveira, edluce.veras, paulo.lopes, barbara.santos, jose.diogo, adriano.oliveira}@sidia.com

ABSTRACT

The growing mobile application development market brings an urgent need for tools capable of evaluating and comparing similarities between screens of different applications. This article proposes a comparative analysis of methods for similarity detection on Android screens, integrating Information Retrieval (IR), embedding models (BGE-M3, Snowflake-Arctic-Embed2, Nomic-Embed-Text) and Vision Language Models (VLM) such as qwen2.5vl:7b and gemma-3:27b. The study addresses challenges like design redundancies and UI test inconsistencies by using combined pipelines: IR, embeddings, and VLM for screen identification. The results demonstrate the strengths and limitations of each approach, providing insights into their applicability in mobile UI similarity detection.

KEYWORDS

Android UI similarity, IR, embedding models, VLM

1 INTRODUCTION

The rapid growth of the mobile application development market has created a pressing need for tools capable of evaluating and comparing similarity between screens from different applications. This is particularly challenging due to the complexity of screen content, which involves both visual elements and textual information.

Detecting this similarity is a complex problem that involves considering multiple factors, including layout design, user interface components, color schemes, and text content.

IR and VLMs have been successfully applied in various areas of Artificial Intelligence, including image classification, object detection, and natural language processing. However, their application in detecting similarity between Android screens remains an ongoing challenge due to the unique characteristics of mobile devices and the variability of screen content.

In this article, we focus on a comparative analysis of methods for detecting similarity in Android screens, integrating IR, embedding models (BGE-M3, Snowflake-Arctic-Embed2, Nomic-Embed-Text) and VLM such as qwen2.5vl:7b and gemma3:27b. Our goal is to investigate the effectiveness of these approaches in detecting similarity between screens and identify potential areas for improvement.

In: Proceedings of the Brazilian Symposium on Multimedia and the Web (WebMedia'2025). Rio de Janeiro, Brazil. Porto Alegre: Brazilian Computer Society, 2025.

ISSN 2966-2753

2 BACKGROUND AND RELATED WORK

2.1 Information Retrieval (IR)

IR is a branch of Computer Science that encompasses data analysis and aims to locate relevant documents within a repository, usually an index, through queries. This can include, but is not limited to, images or texts visually similar to the query made. According to, Zhu et al. (2023), The core function of an IR system is retrieval, which aims to determine the relevance between an user-issued query and the content to be retrieved, including various types of information such as texts, images, music, and more[11]. The challenges found in IR include semantic ambiguities that can have different meanings depending on the context, visual variation in images with poor lighting, images with different angles, or noise that can hinder matching, and the computational cost of maintaining large repositories with millions of images and files.

2.2 Embedding models (BGE-M3, Snowflake -Arctic-Embed2, Nomic-Embed-Text)

Word embedding in simple term can be defined as representing text in form of vectors [10]. In this scenario, BGE-M3 (BAAI General Embedding - Multilingual, Multi-function, Multi-vector) stands out for its structure that can handle multiple languages, multiple functions, and multiple vectors within a single architecture.

According Zhan et al. (2024), M3-Embedding realizes three-fold versatility. It supports a wide variety of languages and handles input data of different granularities. Besides, it unifies the common retrieval functionalities of text embeddings[3].

So far, there have been many methods that can be used for tasks such as classification. However, this model stands out by offering dense and adaptable vector representations that maps queries and documents to distinct vector spaces, enabling scalable approximate search systems.

The Arctic Embed 2.0, developed by Snowflake, is an evolution of the Arctic Embed, designed for high-performance tasks related to Retrieval-based Indexing. They aim to capture nuanced semantic meaning with high fidelity, achieving state-of-the-art performance on benchmarks such as MTEB, BEIR[5].

The Nomic-Embed-Text model was developed by the Nomic AI team with the goal of providing optimized vector representations for tasks such as similarity, clustering, semantic search, and vector indexing. Focusing exclusively on text efficiency, it makes it highly suitable for applications like knowledge graph vector searches, semantic summaries, and text-based content recommendations[8].

2.3 Vision Language model (VLM)

The VLMs are designed to interpret and generate information by integrating visual signals (images and videos) with text, enabling tasks such as answering questions about images.

Among them, qwen2.5vl:7b, developed by Alibaba Cloud, stands out for its significant improvements in instructional alignment, deep semantic understanding of images, and accurate textual production from visual content. According to Bai et al., this method not only reduces computational costs but also provides a flexible way to dynamically compress image feature sequences of varying lengths[1].

Another notable model is gemma3:27b, developed by Google DeepMind for multi-modal tasks, although still in development. The gemma3:27b is being designed to natively support visual inputs with high efficiency in resource-constrained environments. Both models are capable of understanding, inferring, and generating multimodal content, with AI-assisted learning systems [9].

The application of VLM for image description in the literature has proven to be highly useful, they have made progress in the field of user interface analysis by combining visual and textual information[4]. Haque and Csallner introduced IconDesc, a tool that demonstrated significant improvements in generating relevant alt-text, by the use of the Gemini LM3 on UI elements identification [4].

Modi et al., uses the LLM Meta-Llama 3.2-11B-Vision-Instruct model to interpret images and generate a description with relevant accuracy. The LLM was utilized like a part of a structure of an application where the user provides images, which are processed to generate descriptions and captions with accuracies ranging from 80 to 90%. The project demonstrated advancements in image captioning[6].

In Nguyen-Mau et al.'s work, a collection of VLM were used, such as: Qwen2-VL-7B-Instruct, MiniCPM-Llama3-V-2-5 and Llama-3.2-11B-Vision-Instruct. The proposal was to use a dataset with text and images for the VLM to interpret and respond to a questionnaire with the aim of creating a ranking to establish which VLM was better[7].

Chai et al. have used GPT-4o as an aid to build a large dataset called AMMEX (Android Multi-annotation Expo), designed to contribute with GUI agents by delivering multi-level descriptions and detailed elements of Android's GUIs [2].

3 METHOD

The experiment was designed to evaluate the effectiveness of similarity detection methods on Android screens. Two key pieces of information were collected from an Android device: a screen dump and a screenshot. These elements were processed differently: the screen dump underwent filtering to remove unnecessary information, while the screenshot was fed into a VLM for natural language description. The combination of both pipelines consists in the dataset. Once prepared, it is submitted to the vectorization process and can subsequently be used for similarity search. The figure 1 illustrates the solution pipeline.

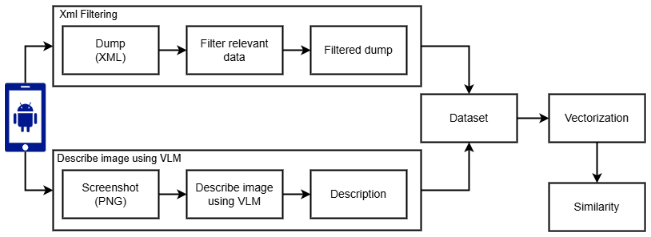

Figure 1: Solution pipeline

4 METHODOLOGY

To conduct the experiments we used the Android Settings Application screen on an Android smartphone. Was collected android's screens, and extracted a XML, that we call it dump, that represents the hierarchy view of Android elements on screen, and a Screenshot to further processing using VLM.

The Python script begins by utilizing Android Debug Bridge (ADB) to extract device dumps and screenshots. Following this, XML data undergoes filtering as detailed in the subsequent section. Each screenshot is then sent to a VLM for description. The resulting texts are combined and vectorized using methods such as TF-IDF or embeddings, with results stored across various indexes.

To simulate small changes that can occur during Android navigation, noise is added to each screen capture or dump before similarity calculations. The input to this process can be either the description of the captured screen or the text extracted directly from the dump. This ensures that even minor interface alterations do not hinder recognition. The noise introduction operates by applying modifications to 10% of the words in the text. For each selected word, one of three actions is randomly performed with equal probability: deletion of the word, insertion of a random word, or substitution of the word with another random word. This controlled introduction of noise assesses the system's robustness. The cosine similarity method is specifically employed in this experiment to measure how well the system can recover and recognize screens, even when they differ slightly.

We use a notebook equipped with a CPU Intel Core i7 and 16 GB of RAM. All the scripts were written in Python 3.12 for processing tasks such as filtering, similarity analysis, and communication with Android devices via the ADB. To interact with VLM and embeddings, we utilized LangChain, an open-source library designed to simplify this type of implementation. Additionally, a local machine running on 42GB RAM and a Nvidia A100 with 40GB VRAM was used to execute Ollama with the gemma3:27b and qwen2.5vl:7b models for image processing tasks and the embedding described.

4.1 Filter relevant data

After the XML is extracted, it is necessary to execute some filtering in order to remove information that could mislead the results. The filter was performed in two steps: 1) Relevant information extraction 2) "Stop words" and special characters removal

4.1.1 Relevant information extraction: The dump XML generated is a document with a large number of special characters, and repeated

Table 1: dataset composition

Screen id	Data Type	Value
0	XML	text
0	gemma3:27b	description
0	XML + gemma3:27b	text + description
0	qwen2.5vl:7b	text
0	XML + qwen2.5vl:7b	text + description
...	...	...
N	XML	text
N	gemma3:27b	description
N	XML + gemma3:27b	xml text + description
N	qwen2.5vl:7b	text
N	XML + qwen2.5vl:7b	xml text + description

terms. The node name, as well as its attributes ("index," "class," "package," etc.), are repeated throughout the document and across the dump of all screens. For that reason, attributes that hold only "true" or "false" values, like "checkable" and "clickable", were discarded. So, at this step, instead of getting all XML content, only the value of the most relevant attributes were collected: "text", "resource-id", "class", "package" and "content-desc".

4.1.2 Stop words and special characters removal: After getting the value of most relevant attributes, the data still contained special characters and some irrelevant words, which we considered as stop words. For this reason, in this stage, these characters and the stop words "com" and "android" were removed, as they did not bring relevance to the data, since they appeared in practically all screens.

4.2 Describe image using VLM

Two VLM were used for comparison purposes: gemma3:27b and qwen2.5vl:7b in this phase. The input to each VLM consisted of a screen image, which was paired with a specific prompt, requesting to describe the image. Was used temperature and top-k hyperparameters with value 0 and 1 respectively. This ensure a minimal variation of image description in experiments. Was selected those hyperparameters experimenting, where it was submitted same image to VLM and choose the hyperparameters with the lowest variation in the output. Was used a python script with langchain to submit the screenshot using Base64 encoding.

4.3 Dataset

The dataset is formed after a pre-processing of XML and Screenshots following the processing outlined in image 1, we generated the input data in two stages: 1) text of most representative field extracted from XML to be vectorized, and 2) The description of the Android Screenshot provided by a VLM. After these steps, we make combination with the result to obtain a total of 5 input data points for N cases presented in table 1.

4.4 Vectorization

After generation of dataset, it is generate 3 groups of outputs that can be combined with different techniques to generate vectors to compute similarity between screens. Exist many forms to transform a text to a numeric vector to be processed by machine, in this study uses the TF-IDF and BGE-M3, Snowflake-Arctic-Embed2, Nomic-Embed-Text embeddings. To generate indexes was used the output of Filter relevant data, Describe image using VLM, and the combination of both as below:

- XML: This generate an index with the output of Filter relevant data step;
- VLM Description: This index uses the output of Describe image using VLM;
- XML + VLM Descrition: This index uses the combination of XML input and VLM Description. In the process the text of both outputs is concatenated and added into embedding.

Table 2: TF-IDF indexes

Model	Input
Vectorizer Technique	xml
Vectorizer Technique	gemma3:27b
Vectorizer Technique	xml + gemma3:27b
Vectorizer Technique	qwen2.5vl:7b
Vectorizer Technique	xml + qwen2.5vl:7b

The Term-Frequency-Inverse-Document-Frequency (TF-IDF) methodology produces five unique indexes. For the embeddings, three different embedding models generate three different sets of five unique indexes each. In total, we will have 4 unique index like represented in Table 2.

4.5 Similarity

After all indexes are generated, we use the cosine formula to calculate the similarity for each screen in each index. This results in a list of matrices of N x N dimensions, where N is the number of screens and each position represents the similarity between two screens. Each line corresponds to the similarity between a screen and all other screens, while the main diagonal contains the similarity of a screen with itself. We extract the main diagonal and store in Set 1 and extract the biggest similarity in each line that is not in main diagonal to Set 2.

5 EVALUATION METRICS

To evaluate the study of the use of VLM, TF-IDF and Embeddings in finding similarity between screens, comparative results of the index were used to extract accuracy, assertiveness, and the ratio of the obtained data. Accuracy is obtained through the similarity algorithm that compare screens using cosine distance, providing a number between 0 and 1, where 1 represents that the screens are identical. Accuracy is extracted from 2 sets of similarity for each test model with different VLM and Embeddings.

Set 1 store the similarity when we compare screens with themselves, aiming to understand the degree of assertiveness of the models for identical screens. Set 2 store the bigger when we compares screens with each other, excluding themselves, aiming to understand the degree of assertiveness in identifying different screens.

With the accuracy of these 2 sets for each model, we can obtain some data used for evaluation:

Table 3: Results Min Max composition

Vectorizer	Data Type	Min	Max	Min/Max ratio
TF-IDF	xml	0.98	0.95	1.03
TF-IDF	gemma3:27b	0.96	0.79	1.21
TF-IDF	xml + gemma3:27b	0.97	0.85	1.14
TF-IDF	qwen2.5vl:7b	0.96	0.82	1.18
TF-IDF	xml + qwen2.5vl:7b	0.97	0.85	1.15
bge-m3	xml	0.70	0.92	0.75
bge-m3	gemma3:27b:27b	0.91	0.91	1.00
bge-m3	xml + gemma3:27b	0.61	0.92	0.66
bge-m3	qwen2.5vl:7b	0.84	0.93	0.90
bge-m3	xml + qwen2.5vl:7b	0.65	0.92	0.70
snowflake	xml	0.47	0.92	0.51
snowflake	gemma3:27b	0.90	0.91	1.00
snowflake	xml + gemma3:27b	0.44	0.91	0.49
snowflake	qwen2.5vl:7b:7b	0.62	0.93	0.67
snowflake	xml + qwen2.5vl:7b	0.46	0.91	0.50
nomic	xml	0.47	0.95	0.50
nomic	gemma3:27b	0.93	0.96	0.98
nomic	xml + gemma3:27b	0.48	0.95	0.51
nomic	qwen2.5vl:7b	0.92	0.97	0.95
nomic	xml + qwen2.5vl:7b	0.47	0.95	0.50

Table 4: Mean results composition

Vectorizer	Data Type	M1	M2	M1/M2 ratio
TF-IDF	xml	0.99	0.21	4.71
TF-IDF	gemma3:27b	0.98	0.21	4.66
TF-IDF	xml + gemma3:27b	0.99	0.18	5.50
TF-IDF	qwen2.5vl:7b	0.98	0.21	4.66
TF-IDF	xml + qwen2.5vl:7b	0.99	0.18	5.50
bge-m3	xml	0.93	0.64	1.44
bge-m3	gemma3:27b	0.96	0.74	1.31
bge-m3	xml + gemma3:27b	0.90	0.62	1.43
bge-m3	qwen2.5vl:7b	0.95	0.70	1.37
bge-m3	xml + qwen2.5vl:7b	0.90	0.62	1.46
snowflake	xml	0.86	0.58	1.49
snowflake	gemma3:27b	0.96	0.60	1.60
snowflake	xml + gemma3:27b	0.82	0.52	1.56
snowflake	qwen2.5vl:7b	0.94	0.58	1.63
snowflake	xml + qwen2.5vl:7b	0.80	0.51	1.58
nomic	xml	0.91	0.70	1.30
nomic	gemma3:27b	0.98	0.80	1.23
nomic	xml + gemma3:27b	0.88	0.68	1.29
nomic	qwen2.5vl:7b	0.98	0.78	1.25
nomic	xml + qwen2.5vl:7b	0.88	0.66	1.33

- The smallest value found in Set 1 (Min): The worst case of the Set 1;
- The largest value found in Set 2 (Max): The worst case of the Set 2;
- The ratio between the two (Min/Max ratio): The ratio of the worst cases, allowing for comparison of the distance between worst cases across each model;
- The average of Set 1 (M1): The closer to 1, the more assertive it was in comparing the same screen;
- The average of Set 2 (M2): The closer to 0, the more assertive it was in comparing different screens;
- The ratio between the two (M1/M2 ratio): The higher it is, the better the model was able to differentiate screens within the model's accuracy reality.

6 RESULTS AND DISCUSSION

In table 3 we can see on Min column the smallest similarity in Set 1, meaning that during navigation in a screen already reached and added to the index, it can be recognized even with small changes. On max column we see the biggest similarity in Set 2, with a greater value more risky to recognize wrongly a screen. the ratio of Min and Max tell us the best approach in the worst scenario.

In table 4 we can see the M1 and M2 that are similar to Min and Max, but instead of the minimum similarity to the same screen and maximum to other screen, the M1 is a mean of all similarity in set 1 and M2 is the mean of Set 2, the ratio tell us with is the best approach in general.

7 CONCLUSIONS

This study presented a comparative analysis of IR, embedding models, and VLM in detecting similarity between Android screens. As shown in Table 3, TF-IDF with "XML + description" was the most effective method for capturing similarities within the same screen, while TF-IDF with generated descriptions stood out in differentiating between different screens, both in the worst-case scenario and within the context of the Settings app, which contains a significant amount of text. Meanwhile, Table 4 confirms that TF-IDF was a better choice for vectorization, followed by Snowflake embedding as the second generally better approach, especially when used with descriptions generated through Qwen2.5vl. The integration of multiple approaches demonstrated greater robustness and accuracy in similarity detection.

Interestingly, the performance of TF-IDF, a classic technique, compared to more modern embedding models warrants consideration. It is hypothesized that this occurs because the task of identifying screens relies more on exact keywords – such as resource-ids or specific text – which TF-IDF effectively captures, while embeddings may smooth out these details in favor of broader semantic meaning.

Future work should focus on developing hybrid models that combine other VLM and VLM hyperparameters not addressed in this study. These findings hold promise for real-world scenarios such as automated testing, accessibility enhancements, and improved user experience analysis on mobile devices.

ACKNOWLEDGMENT

This paper is a result of the R&D&I Project (ASTRO) performed at Sidia Institute of Science and Technology sponsored by Samsung Eletrônica da Amazônia Ltda., using resources under terms of Federal Law No.8.387/1991, by having its disclosure and publicity in accordance with art.39 of Decree No.10.521/2020.

REFERENCES

[1] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923* (2025).

[2] Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Dingyu Zhang, Shuai Ren, and Hongsheng Li. 2025. AMEX: Android Multi-annotation Expo Dataset for Mobile GUI Agents. arXiv:2407.17490 [cs.HC] https://arxiv.org/abs/2407.17490

[3] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216* (2024).

[4] Sabrina Haque and Christoph Csallner. 2024. Inferring Alt-text For UI Icons With Large Language Models During App Development. arXiv:2409.18060 [cs.HC] https://arxiv.org/abs/2409.18060

[5] Luke Merrick, Danmei Xu, Gaurav Nuti, and Daniel Campos. [n. d.]. Arctic-embed: scalable, efficient, and accurate text embedding models (2024). *arXiv preprint arXiv:2405.05374* ([n. d.]).

[6] Aayush Modi, Vrajkumar Patel, Harsh Mistry, Abhishesh Mishra, Rocky Upadhyay, and Apoorva Shah. 2025. From Alt-text to Real Context: Revolutionizing image captioning using the potential of LLM. *International Journal of Scientific Research in Computer Science Engineering and Information Technology* 11 (01 2025), 379–387.

[7] Trong-Hieu Nguyen-Mau, Nhu-Binh Truc, Nhu-Vinh Hoang, Minh-Triet Tran, and Hai-Dang Nguyen. 2025. *Enhancing Visual Question Answering with Pre-trained Vision-Language Models: An Ensemble Approach at the LAVA Challenge 2024*. 281–292. https://doi.org/10.1007/978-981-96-2641-0_19

[8] Zach Nussbaum, John X Morris, Brandon Duderstadt, and Andriy Mulyar. [n. d.]. Nomic embed: training a reproducible long context text embedder (2024). *arXiv preprint arXiv:2402.01613* ([n. d.]).

[9] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786* (2025).

[10] Parul Verma and Brijesh Khandelwal. 2019. Word embeddings and its application in deep learning. *Int J Innov Technol Explor Eng* 8, 11 (2019), 337–341.

[11] Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Haonan Chen, Zheng Liu, Zhicheng Dou, and Ji-Rong Wen. 2023. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107* (2023).