CLEAR: A Fully User-side Image Search System

Ryoma Sato
r.sato@ml.ist.i.kyoto-u.ac.jp
Kyoto University / RIKEN AIP
Kyoto, Japan

ABSTRACT
We use many search engines on the Internet in our daily lives. However, they are not perfect. Their scoring function may not model our intent or they may accept only text queries even though we want to carry out a similar image search. In such cases, we need to make a compromise: We continue to use the unsatisfactory service or leave the service. Recently, a new solution, user-side search systems, has been proposed. In this framework, each user builds their own search system that meets their preference with a user-defined scoring function and user-defined interface. Although the concept is appealing, it is still not clear if this approach is feasible in practice. In this demonstration, we show the first fully user-side image search system, CLEAR, which realizes a similar-image search engine for Flickr. The challenge is that Flickr does not provide an official similar image search engine or corresponding API. Nevertheless, CLEAR realizes it fully on a user-side. CLEAR does not use a backend server at all nor store any images or build search indices. It is in contrast to traditional search algorithms that require preparing a backend server and building a search index. Therefore, each user can easily deploy their own CLEAR engine, and the resulting service is custom-made and privacy-preserving. The online demo is available at https://clear.joisino.net. The source code is available at https://github.com/joisino/clear.

CCS CONCEPTS
• Information systems → Web searching and information discovery; Users and interactive retrieval.

KEYWORDS
Information Retrieval; Web Searching; User-side Systems

ACM Reference Format:
Ryoma Sato. 2022. CLEAR: A Fully User-side Image Search System. In Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM ’22), October 17–21, 2022, Atlanta, GA, USA. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3511808.3557172

1 INTRODUCTION
A massive amount of information is uploaded on the Internet every day, and it becomes more important yet difficult to search for desired information from the flood of information. Thus, many functionalities of search engines have been called for, including multi-modal search [6, 8, 13, 30] and fairness-aware systems [5, 22, 27]. However, many information retrieval systems on the Internet have not adopted rich functionalities, and they usually accept simple text queries only. Even if a user of the service is unsatisfied with a search engine and is eager to enjoy additional functionalities, what he/she can do is limited. In many cases, he/she continues to use the unsatisfactory system or leaves the service.

Let us consider Flickr1, an image hosting service, as an example. Suppose we are enthusiastic users of Flickr. Although there are many amazing photos and communities on Flickr, we are not satisfied with the search engine of Flickr. Specifically, it accepts only text queries only. Even if a user of the service is unsatisfied with a search engine and is eager to enjoy additional functionalities, what he/she can do is limited. In many cases, he/she continues to use the unsatisfactory system or leaves the service.

1https://www.flickr.com/
We hope CLEAR facilitates this trend. We expect many characteristic user-side search systems will appear. We can crawl images from Flickr, build a search index by faiss [11], and host the search engine by ourselves. However, there are more than five billion photos on Flickr, and it typically takes a few weeks to crawl Flickr even if we selectively crawl pages with focused crawling techniques [7, 12, 16, 20]. Besides, it also requires a costly backend server to host a search engine. Therefore, this choice is neither practical for many users.

Another possible solution is to build a search system by ourselves. We can crawl images from Flickr, build a search index by faiss [11], and host the search engine by ourselves. However, there are more than five billion photos on Flickr, and it typically takes a few weeks to crawl Flickr even if we selectively crawl pages with focused crawling techniques [7, 12, 16, 20]. Besides, it also requires a costly backend server to host a search engine. Therefore, this choice is neither practical for many users.

Recently, more practical user-side information retrieval algorithms have been proposed [24, 25]. They run on a browser and do not need a backend server. However, the evaluations of existing works are conducted offline, and it is still not clear if this concept works in practice. Indeed, the official implementation of Tiara [25]², a user-side image search algorithm, requires a few minutes for a single search, which significantly degrades user experiences and is not practical for a real-time search system. In this demonstration, we simplify and optimize Tiara and thereby show a practical user-side image search system online. This is the first practical implementation of a fully-user side search system. We named our system CLEAR, which stands for CIent-side sEARCH. The features of CLEAR are as follows.

Lightweight. As mentioned above and illustrated in Figure 1, traditional search systems require crawling the Web and building a search index, which takes many weeks and consumes much computational and network resources. By contrast, CLEAR does not require initial crawling or indexing, i.e., CLEAR incurs zero overhead. Each user can fork, deploy, and use CLEAR instantly. It is also easy to redefine their own scoring function. Thus, end users can easily enjoy their own search systems.

Fast Iteration. When we update the feature extractor or scoring function, an ordinary system would need to rebuild the index. By contrast, as CLEAR does not use any search indices in the first place, we can seamlessly update the scoring function just by rewriting the JavaScript snippet. It accelerates the development of the desirable scoring function.

Privacy-aware. If we adopted a traditional system configuration with a backend server, we could snoop on the uploaded images in theory. By contrast, as CLEAR runs totally on the user-side, users do not have to worry about privacy issues.

In addition, we stress that user-side systems do not need to be used privately. Rather, each user can publish their own search engines with their own scoring function and their own interface. As CLEAR does not need a backend server, hosting a user-side system is much easier. For example, their own search system can be deployed on a static page hosting service such as Amazon S3 and GitHub Pages. We expect many characteristic user-side search systems will appear on the Internet, and it will become easy to find favorite engines. We hope CLEAR facilitates this trend.

²https://github.com/joisino/tiara

Figure 2: Interface of CLEAR. Upload an image, and then CLEAR retrieves similar images from Flickr. The functionality of CLEAR is ordinary. The highlight lies rather in how it is realized and how easy deployment is. (Left) User interface. (Right) Search results.

Another use case of CLEAR is prototyping by official/unofficial developers. Even if their goal is to build a traditional system with a backend server, CLEAR accelerates the design of the score function. Besides, as CLEAR can be instantly deployed, interface designers can try and develop the system, and they can carry out user studies before the backend engineers complete the system.

2 RELATED WORK

User-side information retrieval systems [23–25] enable each user to build their own system, whereas traditional systems are developed by service developers. For example, private recommender systems [24] and Consul [23] turn recommendation results into fair and/or diverse ones even if the official recommender systems are not fair or diverse. Tiara [25] is the most relevant work to this demonstration. It realizes image retrieval based on user-defined score functions. The critical difference between Tiara and this demonstration is that the original paper of Tiara conducted only batch evaluations. Indeed, the official implementation of Tiara requires a few minutes for a single search on Flickr, which is too slow for real-time demonstration. We optimize the system and realize a real-time user-side search system.

A steerable system [3, 9] also allows users to customize the system. However, the critical difference with user-side systems is that steerable systems are implemented by official developers, and users cannot enjoy this feature if the official system is not steerable in the first place. By contrast, user-side systems turn ordinary systems into steerable ones on the user-side.

Another relevant realm is focused crawling [2, 4, 7, 10, 17]. In contrast to exhaustive crawling, focused crawling aims to retrieve only relevant information. Although these techniques accelerate crawling, they still require several hours to several weeks. Therefore, most users cannot afford to adopt such systems due to time, network, or computational resources. In stark contrast, our system does not require initial crawling at all.

3 CLEAR

3.1 User Interface

The interface of CLEAR is simple (Figure 2). A user clicks the blue button and uploads an image; then CLEAR retrieves similar images from Flickr. The functionality is an ordinary similar image search. We stress that the highlight of CLEAR lies rather in how it is realized. It is remarkable that CLEAR works similarly to an ordinary system with a backend server.
To quantify the effect of using the greedy algorithm, we conducted an offline evaluation using the Open Image Dataset [14]. The experimental setup is the same as in Tiara [25]. Specifically, we use a subset of the Open Image Dataset with 100,000 images as a virtual image repository and use the labels of an image as a set of tags. We use the same score function as Tiara, i.e., the logit of ResNet. We set the query budget as 100 in this experiment. Figure 3 shows that the greedy algorithm slightly degrades the performance while it is 60 times more efficient. Tiara is effective when there is no tight time constraint. As we call for real-time inference, we adopt the greedy algorithm in the demonstration.

### 3.3 Implementation

CLEAR is implemented with the React framework. It adaptively calls `flickr.photos.search` API via the Axios library. This API takes text as input and returns a set of images and their attached tags. CLEAR manages the average score of each tag and queries the tag with the highest average score iteratively. CLEAR shows the returned images in the order of scores. CLEAR’s score function is

\[
 f(x) \overset{\text{def}}{=} \exp(g(x)\top g(s)/1000),
\]

where \( g \) is the MobileNetv2 [21] feature extractor and \( s \) is the source image uploaded by the user. The higher the inner product similarity to the source image, the higher this score is. The feature extractor is implemented with the TensorFlow.js library [28]. To improve the effectiveness and user experience, CLEAR adopts the following strategies.

**Initial queries.** At first, CLEAR does not have any candidate tags. To generate initial queries, CLEAR first obtains class names of the input image using the MobileNetv2 classifier and uses top-\( k \) class names for initial queries. This is in contrast to Tiara, which uses a fixed set of initial tags.

**Parallelization.** As an API call is the bottleneck of wall-time consumption, CLEAR issues several API calls simultaneously and aggregates results. Specifically, CLEAR selects top-10 tags and issues 10 queries at once. The score evaluation is also parallelized. This is in contrast to Tiara, which issues a single query at once.

**Date Specification.** The `flickr.photos.search` API can specify the dates of images, and the default setting is the latest images. We found that the default setting was not effective and reduced diversity. For example, we upload a cat image, and “cat” and “calico cat” tags are the identified best tags. CLEAR queries both “cat” and “calico cat” tags, and the set of returned images would largely overlap each other. To overcome this issue, we randomly set the dates for each tag so that the returned images do not overlap one another.

**Real-time display.** Although CLEAR is efficient, it still takes tens of seconds to complete a single search, which degrades the user experience. To mitigate this, CLEAR shows the intermediate results, and thereby, users can see the first results in a few seconds. This improves the user experience much.

We also note that the search results of CLEAR are stochastic. First, the Flickr API is stochastic because many images are being uploaded and some images are being deleted. Second, CLEAR’s
algorithm itself is stochastic in specifying the dates and selecting tags. As the former one is difficult to remove, we keep the entire system stochastic. The demerit of this choice would be the lack of reproducibility. However, the merit is that a user can try the same query many times and obtain fresh information every time. This improves the chance of serendipitous findings.

3.4 Bespoke System
The source code of CLEAR is available at https://github.com/joisino/clear. An interested user can fork the repository and build their own system. As mentioned earlier, CLEAR is easy to deploy and host because it is a fully-user side system. The score function is defined in src/score.js. A user can rewrite the function and instantly tries their own scoring function. A small change in the score function may change the behavior of the system, and a user may be able to find a search engine that fits his/her preference.

- getFeature function defined in src/score.js computes feature vectors for both the source image and retrieved images. The embedding layer is defined in embeddingName. Users can use other embedding layers for feature extraction.
- embs2score function defined in src/score.js computes scores. Higher is better. This demonstration uses Math.exp(emb1.mul(emb2).sum().dataSync()[0] / 1000). One can try other functions, e.g., the Gaussian kernel Math.exp(-emb1.squaredDifference(emb2).sum().dataSync()[0] / 1000).

Although we focused on a similar image search system in this demonstration, the score function needs not to measure similarity. If one trains a neural network using his/her favorite images so that preferable images have high scores and uses it as the score function, then the resulting system searches for preferable images. If a fairness-aware score function is used, a fairness-aware search system is realized. Another interesting setting is to use a black-box (possibly buggy) neural network as the score function as proposed in Tiara [25]. CLEAR finds a set of images that activate the neural network. Such instances visually show what the neural network at hand represents [18, 26, 29]. Traditional interpretation methods rely on a fixed set of datasets such as ImageNet, but CLEAR can retrieve images from Flickr, which hosts as many as five billion images. In sum, the creativity for the design of the score function is open to each user.

Users can change the target services on which the search system is built by writing a wrapper in src/flickr.js. The requirement for the target service is that it accepts text and returns a set of images. Users can search images from the Internet using their own score functions if Google Image Search is used instead of Flickr.

As CLEAR does not rely on any backend servers or search indices, one can seamlessly use the system after one changes the score function. One can also change the search target from Flickr to other services by writing a wrapper in src/flickr.js.

Note also that the online demo we provide limits the number of queries because many users may use it simultaneously. A bespoke system allows using more query budgets if needed.

Figure 4: Results. (Left) Source images. (Right) Top-8 retrieved images by CLEAR. (Bottom) A failure example. Although the input image is a cat, the retrieved images are snow leopards.

4 DEMONSTRATION
The online demo is available at https://clear.joisino.net. Users can upload images, and the system retrieves similar images from Flickr. The functionality is simple. We stress again that Flickr does not provide an official similar image search system, and therefore users could not enjoy this feature so far. The highlight is that although the author is not employed by Flickr nor has privileged access to Flickr’s server, CLEAR realizes the new feature of Flickr on a user-side.

Results. Figure 4 shows results of the demo program. It can be observed that CLEAR retrieves visually similar images to the source images. The bottom row shows a failure example. Although the input image is a cat, CLEAR confused this image with a snow leopard. Once CLEAR becomes confident to some extent, it is difficult to escape from the failure mode because it adopts the greedy algorithm without explicit exploration. Improving stability while keeping efficiency is an important future direction.

5 CONCLUSION
This demonstration provides the first practical user-side image search system, CLEAR. The proposed system is implemented on the client-side and retrieves similar images from Flickr, although Flickr does not provide an official similar image search engine or corresponding API. CLEAR adaptively generates effective text queries and realizes a similar image search. CLEAR does not require initial crawling, indexing, or any backend servers. Therefore, an end user can deploy their own search engine easily.

ACKNOWLEDGMENTS
This work was supported by JSPS KAKENHI Grant Number 21J22490.
