Software Engineering for Machine Learning Systems Report

Daniyal Shahzad

https://github.com/daniyalshahzad/albumy

Technical descriptions

Two new ML-powered features were implemented to activate as soon as a user uploads a photo, which results in alternative text generation, image caption generation, and image search. These two features allow the description and tag fields of the image to be prepopulated with results generated by ML-based captioning and detection models, respectively, thereby satisfying the requirements of providing automatically generated alternative text and building a mechanism to allow for image search based on objects presented in images.

Alternative text generation

Alternative text generation is achieved by the additional function `get_image_caption(image_path)` at `albumy/blob/master/albumy/blueprints/main.py` (later referred to as `main.py`) lines 29-48. In addition, function `upload()` in `main.py` were modified to include the function call at lines 187 and 190.

After the server receives an image, it will resize the image in two different sizes and save both the image with the original image and the resized images. Then, it will call the function `get_image_caption(image_path)`, with the path of the originally uploaded image. Within the function, the server will open a file called `api.key` provided by users of `albumy`, and read the file to obtain an API key for the replicate library. The function then makes an API call to conduct an inference of Salesforce's BLIP pre-trained LLM, with the user uploaded image as the input, and receives the output of the model, which is a text caption. We resorted to this LLM for its impressive and cost-effective performance as an open-source. It also has inference time and does not rely on high end cloud computes (1 second on an Nvidia T4 GPU). The generated text caption becomes the returning output of `get_image_caption()`, and is used to populate the photo's default description (`photo.description`), as shown in `main.py` at line 190.

Finally, to enable the use of photo's description (`photo.description`) as alternative text, additional modifications of the codebase were needed. More specifically, in `albumy/templates/main/photo.html` at line 12 and `albumy/templates/macros.html` at line 4, in the fields where images are displayed, alternative texts filled with `photo.description` were added as a secondary display option.

Image search

Image search is achieved using tags, an existing mechanism in `albumy` and a property of each photo. However, different from before where tags were defined by users, here additional tags are automatically developed by screening through each uploaded image with an image detector model and obtaining predictions of objects detected from the image. The detections are generated using the `get_image_detection(image_path,photo)` function at `main.py` lines 50-82. Similar to the previous feature, function `upload()` was modified to include the function call at line 195.

After the server receives the image, resizes the image, and populates the image description with the generated caption, it calls the function `get_image_detection(image_path,photo)` with the path of the uploaded image and the photo object as inputs. Within the function, the server makes an API call to conduct an inference of API4AI's object detection model. We resorted to this image detector for its popularity among ML practitioners and cost-effectiveness (at a cost of $0.00035 per inference). The server parses the output of the model, which is a list of detected objects and its associated probability scores, and retains results that have high confidence scores (above 0.5). Subsequently, each of the detected objects is used to associate the photo with a related tag. A query takes place to see if each detected object has been saved in the database as a tag before, and a tag is created if it has not been recorded before. Finally, the property `photo.tags` is populated with the detected objects using `append`. The tags can now be used for image search, as all images related to a certain object can be found by searching via the associated tag.

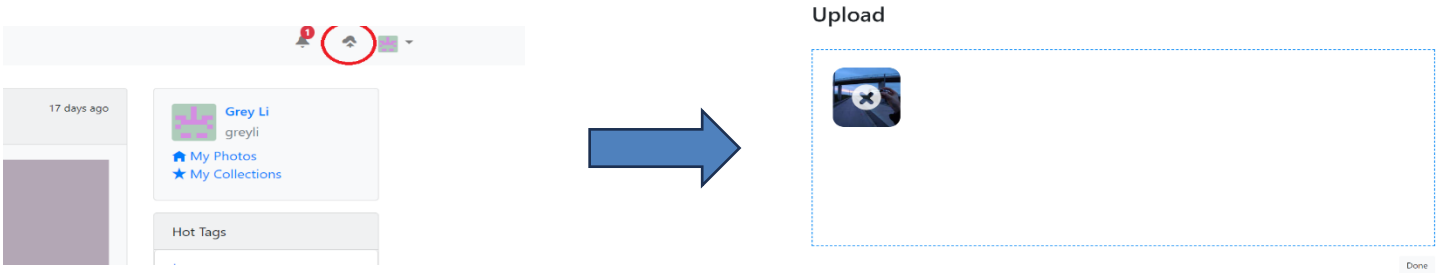
Evaluation of features

For evaluation, we built the demo server using flask and created a fake user profile. We then uploaded twenty images, which contain both personal photos and images from the internet. We then reviewed the uploaded images both on the user page and the home page. We inspected the webpage elements to ensure that the generated captions are displayed as alternative texts. Finally, we ensured that the generated tags for each image are valid and highly accurate.

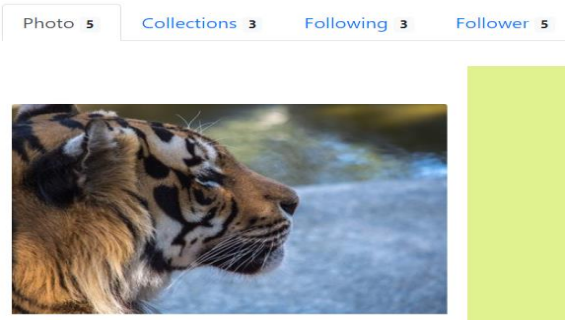
Interface design

The implemented machine learning features are integrated into Albumy, enhancing the user experience without requiring explicit user control.

1. A user begins by clicking the "upload" button in Albumy, located in the top right corner. The User then uploads their photo/s like Albumy without any Machine Learning Features.

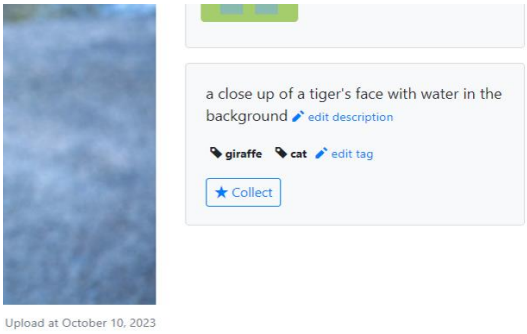


2. Once the user uploads their photos, the photos become part of the database. It is important to note that, up to this point, users won't have the option to add descriptions or tags to the images they are uploading. The user will then be redirected to the page containing their photos as shown in the image.

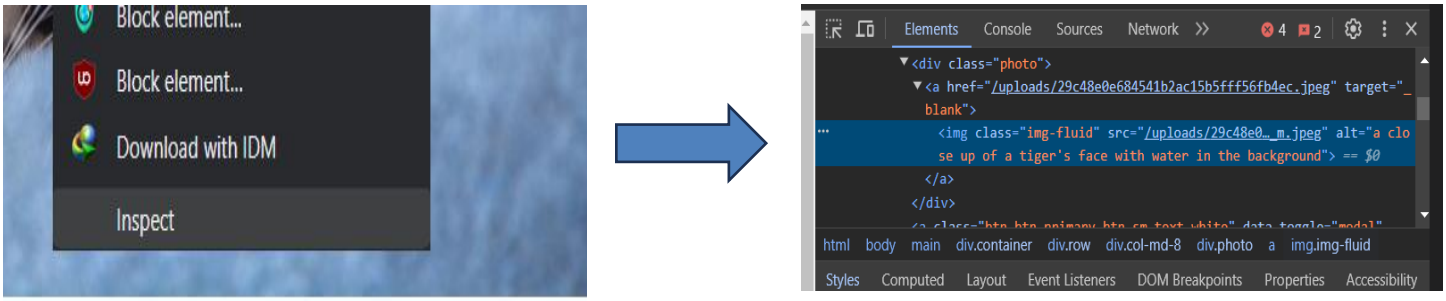


3. This is where the user can click on their uploaded image to view and edit the automatically generated descriptions and tags. For example, the description here read, "A close-up of a tiger's face with water in the background," with tags like "giraffe" and "cat."

   Users can edit their tags and descriptions here, just as they would in Albumy without any machine learning features, by clicking on the "edit description" and/or "edit tag" buttons.



4. Furthermore, if a user right clicks the image and selects inspect, they can view the automatically generated alternate text. It is important to note that this alternative text cannot be edited by the user. Additionally, the alternative text is the same as the automatically generated description.



Production challenges

As with many ML-enabled solutions, the deployment of the proposed ML-features in production would require us to resolve a few technical challenges. Namely, as the website is designed to be a social platform, where monetization is strongly dependent on having a large user base, the additional features should be able to handle millions of users. We believe that our solutions need to address two requirements to become viable as a business product and deliver a satisfying user experience. These two requirements revolve around profitability and perceived inference time.

Profitability

For the social platform to become profitable, it is important that the operating cost is below generated revenue for each user. However, the API calls to obtain results generated from ML models are relatively expensive. In the current implementation, if a user uploads 1,000 images, the caption generation would cost $0.225 USD for the caption generation and $0.350 USD for the tag generation, or $0.575 USD in total. This is a rather expensive cost solely for that one user, compared to the ad revenue that can be generated from the active usage of that user. Thus, for future iterations, one should consider finding a more affordable API or even host its own ML server for model inference. In addition, to cover the additional cost and to build additional revenue streams, one could also consider turning these two ML-enabled features as premium features that are only available to paying users.

Perceived inference time

Users on social platforms look for a seamless user experience with virtually no wait time. However, in the current implementation, after uploading images, the user needs to wait for the image to be successfully uploaded (with resizing, caption generation, and tag generation to be fully complete) to be able to close the upload page. This is not ideal, especially since the ML-enabled features require the inference of both an LLM and an image detector. During the evaluation of our features, we notice that these two models take approximately two seconds in total for inference, and could take even longer when limited computes are allocated for albumy.

To facilitate these ML-powered features and ensure that the uploading process takes virtually no time, it is important to choose cloud compute resources that are scalable and reliable. For example, one can choose to host the inference of the captioning and detection models using AWS EC2 on-demand servers, which scale in terms of memory and compute based on the demand of the jobs requested. In addition, we believe that these ML-enabled features are not time-sensitive features. That is, the users do not necessarily expect them or need them to be generated immediately. For example, in the case of the generation of tags from photos for the purpose of image search, the benefit that it brings is mostly for users to explore photos taken by other users in the past, and search for memories in the photos taken by themselves long before, but not necessarily photos that are immediately uploaded. Thus, we could modify the workflow such that the user can complete the photo upload process and close the uploading page before the model inference is complete. Certain fields for uploaded images, such as description and tags, can be left as incomplete and populated at a slightly delayed time whenever the model inference is complete.

Harms

Alternative text generation and automatic tagging using Machine Learning is a valuable feature of any product as it improves accessibility. It can help users with visual impairment, text-to-speech technology, and low bandwidth networks which are not able to handle relatively large data like images. While these features have significant benefits, they also have some potential harms and challenges.

Issue of privacy

When generating alternative text for images or doing object identification for automatic tagging, the model might inadvertently reveal sensitive information contained within the image, potentially violating the privacy of individuals in the image. In our specific implementation, we generate descriptions, alternative text, and tags for an image without considering user privacy or obtaining their consent. Furthermore, when a user initially uploads a photo, its description, and tags would be automatically generated and applied to the photo. This can become problematic because these features

might introduce privacy concerns for some time until the user can edit the description and tag fields. While users can edit the description field and tags, the alternative text remains independent of user control. These features, as emergent property, have the potential to disclose sensitive information such as credit card details, confidential documents, faces, and locations.

A highly relevant example we want to highlight is Facebook's feature known as "Tag Suggestions," which is analogous to our implementation of tagging in Albumy. Introduced in 2011, it used facial recognition to automatically suggest tags for individuals in photos. While this feature aimed to simplify the tagging process, it raised substantial privacy concerns. Users expressed alarm that their faces could be automatically recognized and tagged without their consent, potentially revealing their presence at specific locations or events. This example illustrates how machine learning-based image recognition has the capability to disclose sensitive information about individuals, their activities, and their locations without their explicit consent.

Furthermore, when users upload images to Albumy, they typically do not expect their personal images to be scrutinized in such a detailed manner, with their content being made available to the public in detail. Consequently, these features may inadvertently disregard users' preferences. These users may not have even consented to their activities being described in alternative text, tags, or the description field.

Lastly, the automatic generation of tags, descriptions, and alternative text can contribute to reinforcing stereotypes. Machine learning models are trained on data that may contain biases against certain groups of individuals, and this automatic generation process can unfairly stigmatize individuals based on their characteristics, potentially breaching their privacy.

Our proposed solution

It is evident that privacy is a significant concern in implementing Machine Learning features. Therefore, we propose several solutions to address this issue. These solutions are derived from the concept of implementing safeguards around the ML model, a topic discussed in the first two lectures. To illustrate this, consider the example of a smart toaster with a simple mechanism like a fuse, that might prevent it from burning the house down. Similarly, we suggest implementing simple heuristics as safeguards around Machine Learning features. More importantly, these safeguards ensure that the privacy of users, particularly those who may not have been attentive enough to configure their privacy controls, is still respected.

The first solution is access control, allowing users to decide how their alternative texts are made available to the public through privacy settings like "private mode," "friends-only," and "public." This also serves as a means for users to provide consent regarding the use of their private data.

Another method to address privacy concerns is by empowering users to edit descriptions, tags, and alternative text. While in our implementation, the users already can edit descriptions and tags, the images are first uploaded and made public with their descriptions, tags, and alternative text automatically generated. This introduces a time window where the user's private information will be made available to the public. Furthermore, the alternative text has remained independent of user control. Therefore, a simple mechanism like the ability to view and edit the automatically generated descriptions, tags, and alternative text before uploading would be handy in solving the issue of privacy.

An additional solution is to replace sensitive content, such as names, with generic English terms. This can be implemented without complex Machine Learning or NLP models since names can be sourced from a database for public content. For instance, "Steven holding a poster" could be replaced with "Someone holding a poster." A similar approach can be applied to replacing or redacting phone numbers, license plates, and other sensitive information in alternative text.

Finally, a simple heuristic to enhance privacy is to limit the word count generated by the ML model for alternative texts. This would result in more generic descriptions instead of highly specific ones. This approach can also be employed for object detection when tagging, giving preference to generic tags already in use rather than highly specific ones.