Vectorized Similarity Search in Multi-modal Data

Shihao Yang
syang536@g.ucla.edu

Haoting Ni
haotinn@g.ucla.edu

Zishun Jin
zishunjin@g.ucla.edu

Yikai Wang
wangyk0817@g.ucla.edu

Yuanxuan Fang
yuanxuanfang@g.ucla.edu

Abstract

As for nowadays real applications, we have many types of data to deal with, such as text, images, audio/video data, html, xml, etc. It's important for us to understand how to deal with multi-model datasets and apply it to recent machine learning classification applications. To do that, we can generate vectors from heterogeneous multi-modal data, including text and images.

1. Introduction

Nowadays, the aim of machine learning and artificial intelligence is to train the datasets we collect under the current condition and predict the trend in the future, doing enormous calculations that human beings will take for a while. Traditional machine learning focus on a single type of data, such as text, images, xml, etc. However, this is sort of violating what a machine can really do because a machine exports images, text, audio,..., and all kinds of data synchronously. It is also like us, we hear, we smell, we feel at the same time. So, to make a machine learn all kinds of datasets at the same time, we need to let the machine analyze multi-model data.

Multi-modal data spans data into different dimensions and different types. Each type might have different features. It seems like combining all those dimensions of data is appealing, but in practice, embedding large scale data is a challenge because it might have noise and conflicts between the dimensions. In this project, we focus on two dimensions of input, text and image. We finally want to achieve that there is a connection between the dimensions, such as if we input a text, we will find the match of its image that best fits the describtion. On the other hand, if we input an image, we will find the match of its image that best describes features of the image.

2. Related Work

Several studies have been conducted on Multi-Modal Queries Through Heterogeneous Network Embedding. In 2022, C. T. Duong et al.[1] proposed an IR system that has been designed to answer genuine multi-modal queries. Compare to baseline techniques, their approach are twice amount of relevant information and also scaling to large multi-model database which is intuitively inspired our method and approach. The model they used to introduce the representation of the heterogeneous data is HINs, which is utilized to capture the semantic relations between data of different modalities of the same entity.

As we are using the COCO Datasets Provided by Microsoft, we also found articles and researchs regarding to the dataset. From the paper by Veit et al.[3] We can get a basic overview of the COCO Dataset and provide us with the thought of how to conduct the Data Exploration step and going forward to process the data into our model.

3. Model Training

3.1. Exploratory Data Analysis

First of all, let's get into the COCO Datasets. From COCO creators: "COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has several features: Object segmentation, Recognition in context, Superpixel stuff segmentation, 330K images (>200K labeled), 1.5 million object instances, 80 object categories. During the process as we

explore the COCO data, we found that there are some useful APIs that can help us explore the data and doing the analysis. We've use the FiftyOne API[2] for data downloading and explore. The following image is from the FiftyOne app and it shows for us that the distribution of the image label.

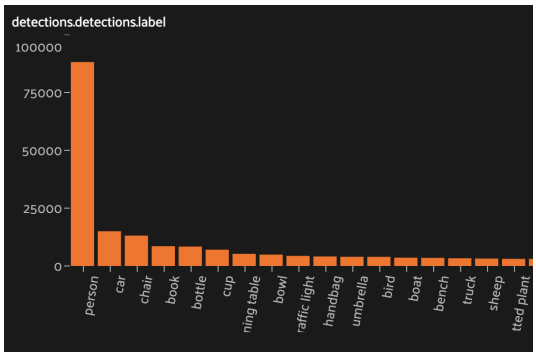


Figure 1: A visualization of our data set used from FiftyOne API.

We can find that the largest number of label of our dataset is from Person which count 88153 among all the image annotations. Car is the second largest label count for 15014 among all the image annotations. It's quite helpful insight for us to see such data structure and help us understand to data better.

3.1.1 Data Structure

To begin our project, we need to deep understand COCO dataset from its inside. All of the COCO data file are in JSON format, which has the dictionary as the top value along with keys and lists nested inside. As we explore, we summarize each layer of the dictionary as follow:

- Info Session: this part is for official use to describe the data with its information like credit and date, etc)

```
{'info': {'description': 'COCO 2014 Dataset',
  'url': 'http://cocodataset.org',
  'version': '1.0',
  'year': 2014,
  'contributor': 'COCO Consortium',
  'date_created': '2017/09/01'},
```

Figure 2: Info session example.

- Licenses section: links to licenses for images in the dataset. One quite worth noticing thing is that its 'id' is directly related to the 'id' in the image section to give them the connections.

```
# del(data['info'])
data.keys()
data['licenses']
```

```
[{'url': 'http://creativecommons.org/licenses/by-nc-sa/2.0/',
  'id': 1,
  'name': 'Attribution-NonCommercial-ShareAlike License'},
 {'url': 'http://creativecommons.org/licenses/by-nc/2.0/',
  'id': 2,
  'name': 'Attribution-NonCommercial License'},
 {'url': 'http://creativecommons.org/licenses/by-nc-nd/2.0/',
  'id': 3,
  'name': 'Attribution-NonCommercial-NoDerivs License'},
```

Figure 3: Licenses example.

- Image section: This is the section we care the most, and its data structure shows as the image below

```
{'license': 1,
 'file_name': 'COCO_train2014_000000247909.jpg',
 'coco_url': 'http://images.cocodataset.org/train2014/COCO_train2014_000000247909.jpg',
 'height': 640,
 'width': 480,
 'date_captured': '2013-11-24 15:11:39',
 'flickr_url': 'http://farm5.staticflickr.com/4126/4841294742_1404700c88_z.jpg',
 'id': 247909},
```

Figure 4: Images example.

As we going through each of its fields: 'license' is the the ID of the image license from the 'licenses' section as we mentioned before. 'file_name' is the the name of the file in the images directory. 'coco_url' and 'flickr_url' are the URLs to the online hosted image copy. 'height' and 'width' can describe the size of image. And finally 'date_captured' which represent when the photo was taken.

- Categories section: this part of the data annotation is using for segmentation stuff and object detection.

```
{'supercategory': 'animal', 'id': 24, 'name': 'zebra'},
{'supercategory': 'animal', 'id': 25, 'name': 'giraffe'},
{'supercategory': 'accessory', 'id': 27, 'name': 'backpack'},
{'supercategory': 'accessory', 'id': 28, 'name': 'umbrella'},
{'supercategory': 'accessory', 'id': 31, 'name': 'handbag'},
```

Figure 5: Categories example.

- Annotation section: This is the most important section of the dataset, which contains important information for each task for different COCO dataset. For every fields inside: 'segmentation' is a list of segmentation mask pixels, 'area' is the number of pixels inside segmentation mask, 'iscrowd' represent whether the annotation is for a single object or multiple (value 0 or 1). 'image_id' which is the 'id' field from the 'images' dictionary. 'bbox' which is the bounding box of the target item in the image. 'category_id' which is the class of the object corresponding to the 'id' in category section.

```
{'segmentation': [[294.19, 50.06, 145.81, 291.61, 52.65, Ellipsis]],
 'area': 64872.04595000001,
 'iscrowd': 0,
 'image_id': 301479,
 'bbox': [29.68, 0.0, 396.32, 502.97],
 'category_id': 18,
 'id': 5850}
```

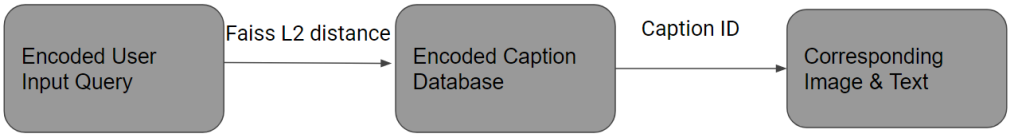
Figure 6: Annotation example.

As for the overall features of the COCO Dataset, we find that there are 80 object categories and 91 stuff categories. Each image has 5 captions for its description. And there are over 250'000 people with 17 different keypoints which are usually used for the human pose anticipation.

3.2. Model Construction

3.2.1 Text As input to Find image Representation

For this NLP project, we will be using Sentence Transformer to encode the input text and the keywords in our database. Sentence Transformer is a natural language processing (NLP) model that can be used to encode text into numerical representations called embeddings. These embeddings capture the meaning and context of the input text, allowing us to compare and analyze the text in a quantitative way. To prepare the materials for embeddings, we will first need to extract the keywords from the annotations of the COCO dataset using the Natural Language Toolkit (NTLK). Each image has 5 annotations so the repatetion may cause overfitting in our later training. NTLK is a popular Python library for working with human language data. This will allow us to easily identify the most important words and phrases in the 5 annotations for each image. We will then divide the dataset into 50 smaller chunks and encode each chunk individually using FAISS. This will allow us to store the encoded data locally and avoid overloading the system with too much data at once. Once all of the chunks have been encoded, we can then combine the encoded data into a single embedding of keywords and their corresponding image ids.

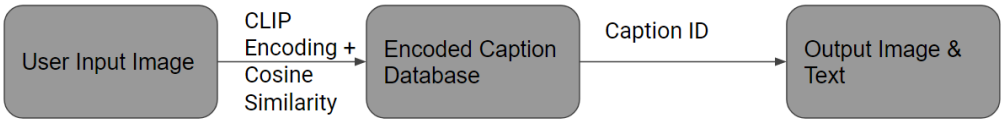


To determine the most related images, we will use the Faiss IndexFlatL2 similarity check to measure the similarity between the input text and the keywords in the database. This will allow us to identify the keywords that are most similar to the input query, and therefore the images that are most related to the input text. By ranking the keywords in order of their similarity to the input text, we can easily locate the images, using their image ids, in the database that are most related to the input query.

3.2.2 Image as Input to Find Text

In this project, we are using the CLIP model to take in an input image and find the most similar images along with their captions in the COCO database. CLIP (Contrastive Language-Image Pre-training) is a large language model trained by OpenAI that can be used to identify and classify images based on their visual content and their associated text captions or labels. To do this, we first input the image into the CLIP model, along with its associated text if available. The model then processes the input using its deep neural network architecture, extracting and analyzing the visual and linguistic features of the image and text.

Once the input image has been processed, the model generates an embedding for the image, which captures the meaning and context of the image in a numerical representation. We can then use this embedding to compare the input image to the images and captions in the COCO database, using the cosine similarity measure to determine the similarity between the input image and the database entries.



Due to the limited VRAM on our system, we have divided the captions in the COCO database into smaller chunks for CLIP to embed. This allows us to encode the data in a more efficient and manageable way, without overloading the system with too much data at once. Once all of the chunks have been encoded, we can then combine the encoded data into a single database of images and their corresponding embeddings. This will allow us to easily compare the input image to the images in the database and identify the most similar entries.

4. Evaluation

For the evaluation part, since we don't know if the standard on the images we get is good or better. First, we will use the Jaccard coefficient (Formula 1) to determine whether the images we get are relevant or not. On here, we not only do the whole sentence check on the input and images' captions but also do the keyword check. If the coefficient is certainly bigger than a threshold=0.8. It will consider relevant. Since the Jaccard coefficient will compare independent words, there is a problem when the input texts are short but the caption is longer, even if the meaning of that text might close, but the Jaccard score might be relatively small. So, we also calculate the cosine similarity (Formula 2) score to check those text similarities that ignore the length of the text. Then we will get the precision at K from the formula which is the relevant image number over K. The reason why we choose k = 10, see figure 8 down below, we have the best precision score at k = 10.

Jaccard Coefficient (Formula 1)

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Cosine Similarities (Formula 2)

$$\cos(A,B) = \frac{A \cdot B}{||A||||B||}$$

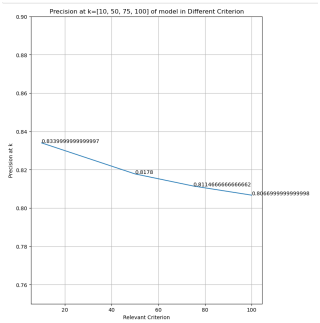


Figure 7: Percision at k with different threshhold.

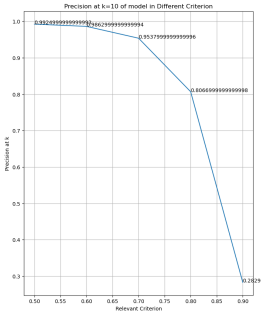


Figure 8: Percision at k with different threshhold.

Also, We tried to plot different precision at k when we use different thresholds (0.5, 0.6, 0.7, 0.8, 0.9), Higher threshold means more relevance between the image and text we searched. Through this plot, we can conclude that when using 0.8 as a threshold and we still have 80 percent precision and we can also filter out texts that are less similar than inputs.

Finally, we calculate the (Normalized discounted cumulative gain) NDCG (Formula 3) to check the desired outputs with the right ranking order. We choose k = 10 as our result numbers, and our NDCG score is relatively high with the input match.

4.1. Software

We majorly used Jupyter Notebook and Google Colab with Python for our project. Here is a list of the libraries that were used in the development of the Python project: numpy, faiss, rake_nltk, nltk, spacy, matplotlib, pycocotools, skimage, pylab, torch, clip, PIL, os. The computer software would be mainly Python through jupyter notebook/Google Colab and we used some packages like Clips, Fifty-One, some OpenAI API, and so on.

4.2. Hardware

We used the default configuration for Google Colab for the lightweight tasks, and we used a desktop with an RTX3070 with 8 GB VRAM and an RTX3080 with 12 GB VRAM for the embedding tasks.

5. Results

In conclusion, we have developed a model that is able to take in an image or text as input and return the 10 most similar images and texts within our COCO dataset. On average, the model is able to produce results within 50 milliseconds, and the output images we tested with various input images and text looked very accurate and visually desirable. In terms of a more rigorous metric, we were able to achieve an accuracy of 81 percent using our own custom evaluation standard. Overall, we are pleased with the performance of our model and believe it has potential for further development and improvement.

6. User Interface

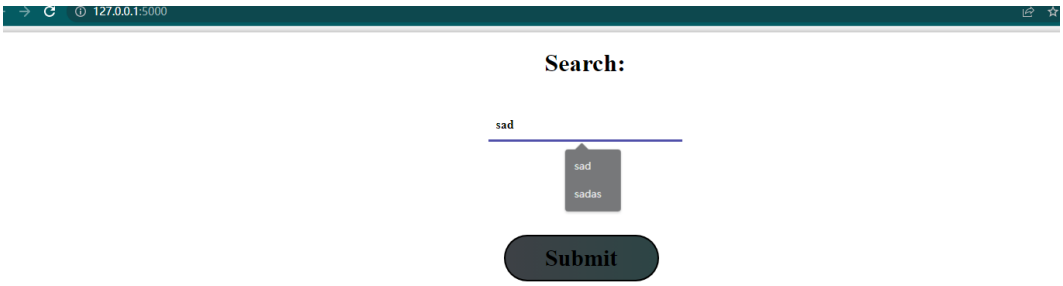


Figure 9: User interface.

The Flask framework is used to create out web application. It is a lightweight Python web framework that provides useful tools and features for building web applications. Flask provides support for interacting with the databases we created and handling user input queries, which is essential for building a functional user interface.

7. Discussion and Potential Application

7.1. Discussion

The evaluation of the returned related images and text in our project is challenging because there is no ground truth data for us to compare against. Without ground truth data, it is difficult to calculate true positive (TP), true negative (TN), false positive (FP), and false negative (FN) rates, which are typically used to compute F1 scores. Therefore, we have had to create our own metrics and evaluation standards to tune and measure the success of our model. These custom metrics and evaluation standards allow us to evaluate the performance of our model and make adjustments as needed to improve its accuracy and effectiveness.

The process of generating embeddings using the CLIP model is very time- and resource-intensive, and our 12 GB GPU is not able to handle the text embedding all at once. To overcome this limitation, we have divided the COCO dataset into smaller chunks and embedded each chunk individually. This allows us to encode the data in a more efficient and manageable way, without overloading the system with too much data at once. Once all of the chunks have been encoded, we can then combine the encoded data into a single database of images and their corresponding embeddings. This will make it easier to deploy the model for use in a production environment, as we will not have to deal with the large amounts of data that would be required to encode the entire COCO dataset all at once.

The accuracy of the results produced by our model is highly dependent on the quality and relevance of the input text. If the input text is ambiguous and even hallucinated, then the model's predictions may be inaccurate or irrelevant. For example, if the input text is "a rainbow alien riding a pig" and we do not have an alien image of that scene in our COCO database, then the model's output may be very inaccurate. To improve the accuracy of the model's predictions, it is important to provide clear and descriptive input texts. Additionally, we can also fine-tune and train the model on a larger dataset of labeled images and text to improve its performance and accuracy.

7.2. Potential Application

Here are some examples of potential uses for this project:

Image search on an e-commerce platform: An e-commerce platform could use the model to enable users to search for products using images instead of text. For example, a user could take a picture of a piece of clothing they like and use the model to search for similar items on the platform. This could be useful for users who are not sure how to describe the items they are looking for, or for users who are looking for items that are similar to ones they already own.

Text-to-image synthesis for social media: A social media platform could use the model to generate images based on the text in users' posts. For example, a user could write a caption for a photo they want to share, and the model could generate the photo based on the caption. This could be useful for users who want to share content that is visually appealing, but don't have access to the actual images they want to use.

Image-to-image synthesis for image editing: An image editing software could use the model to generate new images based on the content of existing images. For example, a user could edit an image by selecting a portion of the image and using the model to generate a new image that is similar to the selected portion. This could be useful for tasks such as image retouching, image restoration, and image enhancement.

Content-based image retrieval for photo sharing: A photo sharing platform could use the model to enable users to search for and share photos that are similar to ones they already have in their collections. For example, a user could upload a photo to the platform and use the model to search for other photos that are similar to the one they uploaded. This could be useful for users who want to share photos with others who have similar interests or aesthetic preferences.

8. Acknowledgements

We gratefully appreciate the help and guidance from our professor Quanquan Gu.

References

[1] C. T. Duong, T. T. Nguyen, H. Yin, M. Weidlich, T. S. Mai, K. Aberer, and Q. V. H. Nguyen. Efficient and effective multi-modal queries through heterogeneous network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 34(11):5307–5320, 2022.

[2] B. E. Moore and J. J. Corso. Fiftyone. *GitHub. Note: https://github.com/voxel51/fiftyone*, 2020.

[3] A. Veit, T. Matera, L. Neumann, J. Matas, and S. Belongie. Coco-text: Dataset and benchmark for text detection and recognition in natural images. page 305–312, 2016.