

# **Capturing and Analyzing Variability in High-Level Image Semantics by Combining Image Annotations with Linguistic Analysis Final Report**

**Alperen Duyan**

## **1) Problem Definition**

The fundamental problem addressed by this project is the subjectivity and variability in the semantic interpretation of images, which is one of the main challenges to effective image-based querying. A single image can be described differently by different people. This causes a Semantic Gap between the image's raw content and the high-level concepts a user might employ for search. To tackle this, the project proposes a methodology that systematically gathers this semantic variability. First, creating an image library by using Generative AI and then collecting subjective annotations from various people. The core objective is to move beyond simple keyword matching by performing linguistic analysis on these two distinct sets of textual data. This analysis will result in the construction of a tri-partite graph, linking the generation prompts, the images, and the resulting subjective annotations. By analyzing the properties of this graph, the project aims to find out the semantic connection of different annotations from different people. Finally, to assess the success of the proposed method by evaluating its performance on a larger dataset.

## **2) Dataset**

In this project, there are two separate datasets. The first one is the dataset which is used to apply text analysis techniques to get semantic relationships between annotations from different people. For this dataset, 30 images are created by using Stable Diffusion and Adobe Firefly. For each generated image, I collected annotations from 3 different people. Together with the generation prompt each image has 4 descriptions in total. To validate the effectiveness of proposed method, it is tested in a larger Dataset. For this purpose, COCO dataset is used. COCO dataset has around 45k images and each image has 5 different annotations.

## **3) Proposed Methodology**

This project's methodology is designed to systematically analyze and map the relationship between an image's generation prompts and its varied human interpretations. The overall process is structured into four main phases. First, an image collection is built using a

generative AI model, ensuring that each image is associated with an initial generation prompt. Second, we introduce variability by having a group of people independently annotate these same images, capturing the range of perceived semantics. Third, a foundational linguistic analysis is applied to both the generative prompts and the subject annotations, utilizing techniques like stop word removal and key term extraction to isolate the core concepts. The final and central phase is the construction of a Tri-Partite Graph. The first level of this tri-partite graph is image annotations, second level is images, third level is human annotations. By analyzing this graph, I try to capture semantic variability and use this for image retrieval. And to evaluate the performance, this relationship is reflected to COCO dataset.

### **3.1) Text Preprocessing**

The first step of text preprocessing is making sure that text is all lowercased. Next, stop words are removed in order to capture only meaningful descriptive words. To do this, NLTK English stop words list is used. Then, words are lemmatized by NLTK Wordnet Lemmatizer. After these steps, the remaining text is tokenized. Every word is a single token.

### **3.2) Vectorization**

To convert the textual data, both the generative prompts and subject annotations, into a numerical format suitable for machine learning, I used the Term Frequency-Inverse Document Frequency (TF-IDF) vectorization scheme. This method calculates a numerical weight for every term in the entire corpus, reflecting how important that term is to a specific document (i.e., a specific image's caption or annotation) relative to the entire collection.

TF-IDF weight has two parts. It is multiplication of Term Frequency(TF) and Inverse Document Frequency(IDF). Term Frequency (TF) measures the local importance of a word, calculated as the count of times a specific term appears. Inverse Document Frequency (IDF) measures the global importance of a word across the entire image collection. It is calculated by taking the inverse of the number of unique annotations (documents) in which that term appears. Therefore, a term used in many different image annotations will have a low IDF score. High TF-IDF scores are assigned to terms that appear frequently in a document but rarely across the collection, highlighting distinguishing concepts. The output of this stage is a high-dimensional, sparse matrix where each row represents an image's combined textual description, and the columns represent the vocabulary terms. This process successfully translates the qualitative, varying semantics of the image

annotations into a quantitative vector space, providing the necessary foundation for subsequent analysis and similarity calculations.

### **3.3) LSA**

After vectorization, the resulting high-dimensional, sparse TF-IDF matrix was processed using Latent Semantic Analysis (LSA). LSA utilizes Singular Value Decomposition (SVD) to decompose the sparse TF-IDF matrix, effectively reducing the number of dimensions while preserving the most semantically meaningful relationships between terms and images. In this project, I specifically reduced the dimensionality to 15 latent topics. This reduction serves two crucial purposes: first, it makes similarity calculations faster and more efficient. Second, it helps to deal with synonymy. By projecting the high-dimensional term space onto 15 core semantic topics, LSA allows images to be compared based on the concepts rather than the exact keywords they contain. The resulting 15-dimensional vectors form the final semantic space used for all image retrieval and similarity computations.

### **3.4) Similarity Computation**

After applying LSA, the resulting matrix is a 120 X 15 matrix. Each column represents a topic which includes several words in it. Each row represents an annotation for an image. Each image has 4 annotations, and 30 images result in 120 rows in the end. I calculated the general representing vector of an image by taking the average of 4 annotations since each annotation has the same importance value. After this step, I have a 30 X 15 matrix. Each row belongs to one image, and each column belongs to a topic. Now these rows are ready for similarity calculations. The method used to calculate this similarity is cosine similarity. Cosine similarity measures the cosine value between two vectors. Bigger cosine values correspond to smaller angles, which means more similarity. After getting the most similar 4 vectors in terms of cosine similarity, I measured Euclidean distance so that I can see how different the annotation of the image retrieved and the mean vector which represents the input image.

## **4) System Implementation**

### **4.1) Architecture Overview**

The Image Retrieval System is designed as a unified application where the **Flask server (app.py)** serves as the central processing unit which is integrating the user interface with the text analysis capabilities. The architecture is logically divided into client-side and server-side components. The client-side (the user's browser running index.html) handles the presentation and interactivity, dispatching user commands (text queries or image

clicks) via asynchronous JavaScript requests. The server-side component, initiated upon application launch, first performs the crucial step of loading all precomputed data and models (TF-IDF vectorizer, LSA model, and image vectors) into memory. I prefer this way because I did not want to repeat all these computations for all requests.

## 4.2) Data Preparation

To increase the performance, I precomputed resource-intensive data structures. The complete set of image annotations was vectorized using the fitted TF-IDF model, and this high-dimensional result was then transformed using the fitted LSA model. The final, reduced-dimensionality vectors for all images (`coco_lsa.npy`) were saved in separate files alongside the vectorizer and LSA models themselves (`tfidf.pkl`, `lsa.pkl`). Upon the Flask application's startup, these assets are loaded directly into memory using `numpy.load` and `joblib.load`. This approach completely bypasses the need to re-transform the entire dataset on every query, enabling the application to respond to user searches instantly by performing similarity calculations directly on the pre-processed LSA vector space.

## 5) Results and Evaluation

### 5.1) Variance of Image Annotations

0.0035	0.0013	0.0009	0.0005	0.0004	0.0000	0.0008	0.0143	0.0031	0.0022	0.0028	0.0001	0.0001	0.0070	0.0031
0.0002	0.0005	0.0025	0.0002	0.0032	0.0000	0.0000	0.0025	0.0028	0.0003	0.0001	0.0002	0.0000	0.0007	0.0003
0.0048	0.0032	0.0000	0.0078	0.0008	0.0000	0.0000	0.0001	0.0001	0.0001	0.0003	0.0000	0.0000	0.0011	0.0009
0.0021	0.0029	0.0002	0.0039	0.0010	0.0001	0.0000	0.0020	0.0065	0.0028	0.0050	0.0001	0.0003	0.0181	0.0030
0.0089	0.0044	0.0000	0.0190	0.0021	0.0000	0.0001	0.0001	0.0001	0.0000	0.0009	0.0000	0.0000	0.0000	0.0004
0.0000	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0004	0.0035	0.0001	0.0001
0.0000	0.0000	0.0003	0.0000	0.0002	0.0000	0.0001	0.0013	0.0010	0.0075	0.0001	0.0004	0.0006	0.0001	0.0002
0.0005	0.0011	0.0077	0.0001	0.0006	0.0000	0.0001	0.0044	0.0084	0.0039	0.0079	0.0004	0.0001	0.0011	0.0028

<b>0.0002</b>	0.0010	0.0121	0.0000	0.0002	0.0000	0.0000	0.0004	0.0012	0.0005	0.0008	0.0004	0.0015	0.0013	0.0022
<b>0.0000</b>	0.0002	0.0000	0.0000	0.0003	0.0000	0.0000	0.0018	0.0007	0.0003	0.0017	0.0000	0.0000	0.0008	0.0010
<b>0.0000</b>	0.0000	0.0000	0.0002	0.0025	0.0000	0.0003	0.0029	0.0014	0.0000	0.0003	0.0022	0.0002	0.0003	0.0001
<b>0.0114</b>	0.0045	0.0000	0.0196	0.0021	0.0000	0.0000	0.0006	0.0006	0.0003	0.0008	0.0001	0.0000	0.0043	0.0061
<b>0.0002</b>	0.0004	0.0230	0.0003	0.0002	0.0000	0.0001	0.0047	0.0029	0.0123	0.0000	0.0003	0.0004	0.0000	0.0004
<b>0.0000</b>	0.0000	0.0000	0.0002	0.0031	0.0000	0.0003	0.0046	0.0014	0.0000	0.0001	0.0002	0.0002	0.0002	0.0003
<b>0.0275</b>	0.0073	0.0005	0.0063	0.0007	0.0001	0.0000	0.0005	0.0008	0.0003	0.0008	0.0000	0.0000	0.0001	0.0047
<b>0.0000</b>	0.0000	0.0000	0.0000	0.0000	0.0094	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
<b>0.0025</b>	0.0006	0.0009	0.0041	0.0050	0.0001	0.0001	0.0135	0.0128	0.0058	0.0047	0.0006	0.0002	0.0112	0.0083
<b>0.0000</b>	0.0001	0.0000	0.0003	0.0044	0.0000	0.0005	0.0071	0.0014	0.0024	0.0005	0.0005	0.0006	0.0003	0.0004
<b>0.0027</b>	0.0011	0.0001	0.0021	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001
<b>0.0029</b>	0.0015	0.0004	0.0000	0.0004	0.0003	0.0001	0.0073	0.0081	0.0045	0.0020	0.0000	0.0002	0.0008	0.0103
<b>0.0008</b>	0.0001	0.0000	0.0030	0.0004	0.0001	0.0000	0.0009	0.0014	0.0005	0.0026	0.0000	0.0000	0.0127	0.0001
<b>0.0005</b>	0.0011	0.0001	0.0001	0.0003	0.0000	0.0003	0.0073	0.0017	0.0031	0.0006	0.0004	0.0001	0.0012	0.0027
<b>0.0000</b>	0.0000	0.0000	0.0001	0.0013	0.0000	0.0082	0.0035	0.0018	0.0001	0.0000	0.0026	0.0003	0.0007	0.0002

<b>0.0000</b>	0.0000	0.0000	0.0001	0.0002	0.0000	0.0003	0.0003	0.0002	0.0000	0.0002	0.0003	0.0001	0.0006	0.0005
<b>0.0000</b>	0.0005	0.0203	0.0000	0.0002	0.0000	0.0000	0.0009	0.0005	0.0012	0.0000	0.0001	0.0001	0.0005	0.0009
<b>0.0024</b>	0.0026	0.0014	0.0002	0.0005	0.0002	0.0000	0.0009	0.0029	0.0013	0.0001	0.0000	0.0001	0.0031	0.0149
<b>0.0000</b>	0.0000	0.0000	0.0000	0.0000	0.0005	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
<b>0.0000</b>	0.0000	0.0000	0.0000	0.0002	0.0000	0.0074	0.0003	0.0002	0.0001	0.0000	0.0001	0.0000	0.0000	0.0000
<b>0.0008</b>	0.0028	0.0002	0.0000	0.0004	0.0106	0.0028	0.0004	0.0023	0.0114	0.0003	0.0001	0.0000	0.0004	0.0009
<b>0.0003</b>	0.0011	0.0006	0.0004	0.0059	0.0000	0.0001	0.0008	0.0013	0.0004	0.0015	0.0000	0.0001	0.0033	0.0014

Table1: Variances for images for 15 topics

I measured the variances of annotations for each image. Since these images are simple and easy to interpret, the variances for images are really small(ranges from 0 to 0.02). This shows that all annotators gave similar annotations for images.

## 5.2) Similar Image Retrieval Results

I have an HTML page which contains our generated images and when we click on one of the images, the page shows us the most similar 4 images.

### Select an Image to Retrieve Similar Images

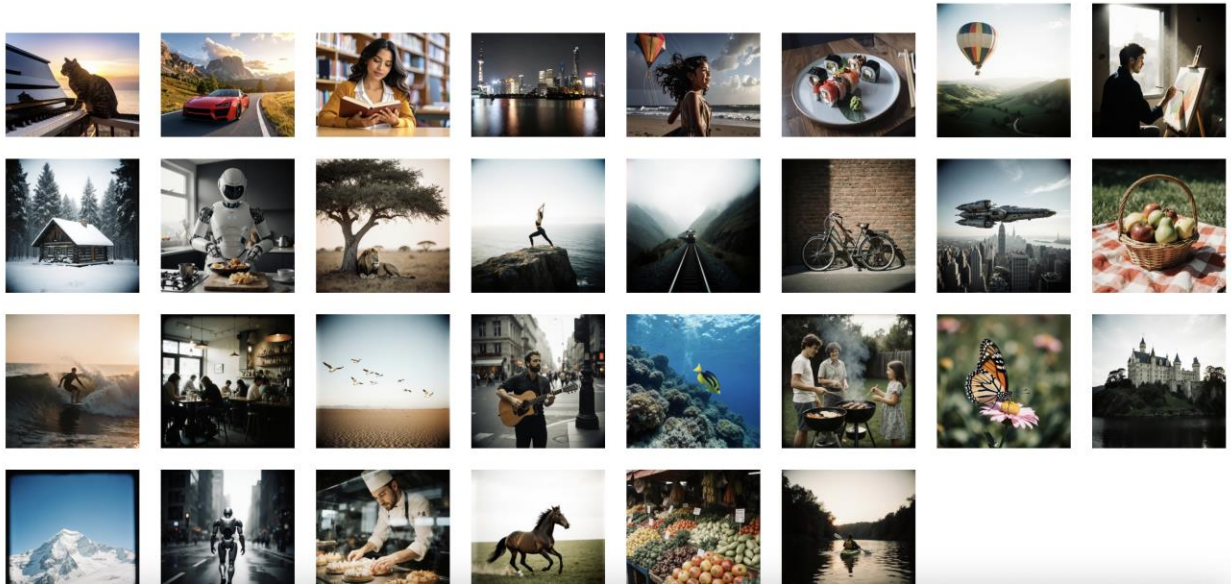


Figure1: HTML page layout

### Results



Figure2: Results when we click on the third image in which a girl reads a book

Cosine Similarities = (0.9985, 0.9963, 0.9963, 0.9963 )

Euclidian Distances = (0.0619, 0.0954, 0.0954, 0.0954)

## Results

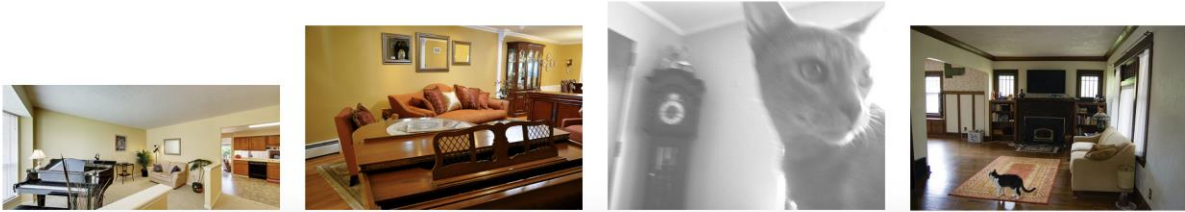


Figure3: Results when we click on the first image (A cat plays a piano)

Cosine Similarities = (0.9976, 0.9970, 0.9970, 0.9970 )

Euclidian Distances = (0.6026, 0.5594, 0.5594, 0.5594)

Obviously the second retrieval is less succesful since we wanted a cat playing a piano but in the retrievals, we either see only piano or only cat images. Also, we can notice that Euclidian distance increases significantly comparing to the first retrieval (nearly 10 times higher). This shows that taking Euclidian distance into account is a good method to evaluate our retrieval success.

### 5.3) Text Search Retrieval Results

In my HTML page, there is also a text box in which we can do text searches.

#### Search

a man playing his guitar

#### Results



Figure4: A succesful text based retrieval

## Search

a watermelon on a plane

## Results



Figure5: An unsuccessful text based retrieval

The success of the retrieval relies on our initial corpus. If we have those words in our corpus, we can have a successful retrieval otherwise we don't. This can be a weakness of this system.

### 6) Conclusion

This project successfully implemented a functional, full-stack Image Retrieval System capable of identifying semantically related images based on both text queries and query-by-example (image-to-image) retrieval. By employing a rigorous pipeline involving textual preprocessing, TF-IDF vectorization, and dimensionality reduction via Latent Semantic Analysis (LSA), I effectively translated the variable human annotations into a robust, low-dimensional semantic space (15 topics). The use of precomputation and the Flask micro-framework ensured that all complex matrix operations were handled efficiently, resulting in a responsive web application where similarity searches using Cosine Similarity are executed with low latency.

### 7) Future Work

**Increase the Number and Variety of Generated Images:** Our initial data is so small and it restricts the function of this approach. Adding more images and more annotations can increase the diversity of our corpus so we can retrieve more different variety images. Also, we can have more difficult to interpret images so that we can increase the variance of annotations which may help us assess the robustness of our method for more complicated images.

**Advanced Embedding Models:** Replacing LSA with more sophisticated deep learning techniques, such as Doc2Vec or leveraging pre-trained vision-language models like CLIP (Contrastive Language-Image Pre-training), could capture better semantic relationships since TF-IDF does not consider the word order relationships.

## **8) Github Repository**

[https://github.com/alperenduyan/Image\\_Retrieval\\_System\\_via\\_Annotations.git](https://github.com/alperenduyan/Image_Retrieval_System_via_Annotations.git)