task1_2

November 2, 2023

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[ ]:
```
vecs_array = np.load("not_own_dataset/vecs.npy", allow_pickle=True).item()
```

[ ]:
```
def get_two_arrays(position, vecs_array, num_of_samples = -1):
    keys_string = str(position) + "_pos"

    if keys_string not in vecs_array:
        return None, None


    print(np.array(vecs_array[keys_string][0]).shape)


    labels = []
    embeddings = []
    sample_counter = 0
    for key in vecs_array[keys_string]:

        for value in vecs_array[keys_string][key]:
            if sample_counter >= num_of_samples and num_of_samples != -1:
                break
            labels.append(key)
            embeddings.append(value)
            sample_counter += 1


    return np.array(embeddings), np.array(labels)
```

[ ]:
```
def randomize_array(embeddings, labels):
    indices = np.arange(len(labels))
    np.random.shuffle(indices)
    return embeddings[indices], labels[indices]
```

[ ]:
```
def print_vecs(data, title, num_to_display=5):
    embeddings, labels = data
    print(f"=== {title} ===")
```

```
    print(f"Displaying {num_to_display} out of {len(embeddings)} data points.")

    # Header
    print("\n{:<10} | {:<20} | Embedding Values".format("Index", "Label"))
    print("-" * 60)

    for idx in range(min(num_to_display, len(embeddings))):
        # Check if embeddings are scalar or array-like
        if np.isscalar(embeddings[idx]):
            truncated_embedding = str(embeddings[idx])
        else:
            truncated_embedding = ", ".join(map(str, embeddings[idx][:5])) + "..
↪."

        print("{:<10} | {:<20} | {}".format(idx, str(labels[idx]),␣
↪truncated_embedding))

    print("\n")
```

In the function get_two_arrays, the third variable is to limit the amout of embedds that gets returned for easier debugging

[ ]:
```
embeddings, labels = get_two_arrays(1, vecs_array, 10)
print_vecs((embeddings, labels), "Before", 10)

random_embeddings, random_labels = randomize_array(embeddings, labels)
print_vecs((random_embeddings, random_labels), "After", 10)
```

```
(122, 1024)
=== Before ===
Displaying 10 out of 10 data points.

Index      | Label                | Embedding Values
------------------------------------------------------------
0          | 0                    | -0.0, -0.0, -0.0, -0.0, -0.0…
1          | 0                    | -0.0, -0.0, -0.0, -0.0, -0.0…
2          | 0                    | -0.0, -0.0, -0.0, -0.0, -0.0…
3          | 0                    | -0.0, -0.0, -0.0, -0.0, -0.0…
4          | 0                    | -0.0, -0.0, -0.0, -0.0, -0.0…
5          | 0                    | -0.0, -0.0, -0.0, -0.0, -0.0…
6          | 0                    | -0.0, -0.0, -0.0, -0.0, -0.0…
7          | 0                    | -0.0, -0.0, -0.0, -0.0, -0.0…
8          | 0                    | -0.0, -0.0, -0.0, -0.0, -0.0…
9          | 0                    | -0.0, -0.0, -4.2848642e-18, -0.0, -0.0…


=== After ===
Displaying 10 out of 10 data points.
```

Index	Label	Embedding Values
0	0	-0.0, -0.0, -0.0, -0.0, -0.0…
1	0	-0.0, -0.0, -0.0, -0.0, -0.0…
2	0	-0.0, -0.0, -0.0, -0.0, -0.0…
3	0	-0.0, -0.0, -0.0, -0.0, -0.0…
4	0	-0.0, -0.0, -0.0, -0.0, -0.0…
5	0	-0.0, -0.0, -0.0, -0.0, -0.0…
6	0	-0.0, -0.0, -4.2848642e-18, -0.0, -0.0…
7	0	-0.0, -0.0, -0.0, -0.0, -0.0…
8	0	-0.0, -0.0, -0.0, -0.0, -0.0…
9	0	-0.0, -0.0, -0.0, -0.0, -0.0…