MMA Invention Assignment

Video Shazam - Query Optimization Through Signatures

Izzy van der Giessen, Vasil Dakov

Abstract

The goal of this invention assignment is to create a Shazam-like system for videos so that a user can give a query video and gets as a result the name of a video similar to it together with the time frame which matched the query video. Our implementation of this system contains an innovation to test the hypothesis described below.

Hypothesis

Precomputing a subset of the data can greatly improve on the speed of the video queries without sacrificing a (substantial) amount of accuracy.

Link between the user perspective and the technical solution

The biggest convenience of Shazam is the way it can quickly find the source of the audio it is presented no matter at which you point the user starts recording. As a user of Video Shazam, I want to have my query return an output (even one that might be slightly off) in a reasonable time. An issue with the sliding window system from Lab 5 was how slow the feature matching was for an entire video (due to the sliding window method it used). The team wants to maximize the speedup of the pipeline by pre-computing signatures and minimize the expected decrease in the accuracy of the final product. This will further satisfy users by making the query experience more seamless.

Motivation of the proposed solution (why is it, in theory, better than alternatives?)

A feature computation over every frame of every video in the database is excessive. While it does undeniably increase accuracy, there is typically no substantial difference between two (or three, or four for that matter) subsequent frames. Thus, due to the intensity of the calculations, it would make sense to try to limit the amount we compute. The proposed method would have the pipeline's most intensive calculations done in advance. The comparison will be instead done through a "signature" on the entire video and multiple small segments over the entirety video, instead of through a frame-by-frame sliding window. In theory, this solution should save the majority of the computation and make the Video Shazam experience more seamless. All that would be left is the computation of the desired feature/s on the input video and the comparison with each precomputed signature. Our goal is not to increase accuracy, but rather to increase speed while sacrificing as little accuracy as possible.

Innovation beyond the material learned in the labs

The way the team came at this innovation was frustration with the speed of the queries in Lab 5. The queries were functional, still based on color histograms, MFCCs, audio powers, and temporal differences.

Waiting more than a minute (in some cases even more) makes the system practically unusable. This is what led to the realization that most of those computations are redundant and gave rise the idea of pre-computing signatures. From then on, the team was focused on minimizing the drawbacks of this new approach.

The first issue that came up was that of the differences between segments in the same video. While it is a fair assumption that a single video has a certain "average" look, many videos have small clips/segments of it that may look radically different. Examples are scene changes in a movie or a cut in a news report. If a user inputs data from just one of these segments, a single signature for the entire video will not be sufficient. The solution the team came up with was segmenting every video in the database into 5-second chunks. The pipeline will use them for the identification of the video segment. That way even when a user gives an arbitrary and seemingly unique chunk of a video, the pipeline has a chance of finding it faster than the aforementioned brute-force method.

Another issue that the team expects is that of a user inputting a video segment between two or more of the 5-second chunks in our database (e.g. 00:03 - 00:08 or 00:03 - 00:17 instead of 00:05 - 00:10). A possible solution is computing signatures for each pair of 5-second segments (so a 10-second signature in this case). This can be extended for queries spanning multiple 5-second chunks by having 10-,20-, and 30-second chunks, building a bottom-up tree structure of video signatures. However, that extension has the drawback of bloating the database size even more and has been left as a point of future improvement.

Quantitative evaluation and qualitative analysis of when it works and doesn't work and why.

The team will focus on three criterion for evaluating our system. Those will be:

1. **Correct video identification**: Did the pipeline manage to find the correct video at all? The output of the pipeline is a list of the five best matches. If the query video is anywhere in those five matches, the criterion is considered satisfied.

2. **Correct segment identification**: Did the pipeline manage to find exactly which part of the videos in the database it was given? The team will consider any intersection with the correct segment a success.

3. **Time taken**: Did the signature method manage to output an answer any faster than the brute-force sliding window method? The criterion is considered satisfied if the speed improvement is more than double.

Based on the combination of these three factors the team will consider whether the system is a success or not. The expected results are faster performance with a decrease in accuracy (due to the nature of limiting the amount comparisons possible). If that decrease in accuracy is substantial enough, even if the system is faster, it would be deemed inefficient. The team defines a *substantial enough* decrease as the signature pipeline failing to identify a video the brute-force pipeline manages to find. This issue should be fixable by adding more data.

Pitch presented during the Lab

As was clear during the labs, a brute force way of comparing videos frame by frame is too slow for a usable Shazam-like system. This is why the team decided to innovate on the way videos are compared by creating a signature for every full video and every five second segment. The first step in the pipeline, after pre-processing, is to compute a signature for the query video. Instead of comparing every frame in video segments, only the signature is compared and the k best results are then used to find the best match of the query video. Our goal is to improve performance without decreasing accuracy. This is why time is our main evaluation metric, which we examine quantitatively. A shorter run time without a noticeable decrease in accuracy is considered a success. As said in the previous section,

Assumptions made

1. **Perfect Localization** (still video in video, but well-localized). The innovation wants to abstract from the variability of localization to test the speed improvement as much as possible.
2. **The average user video segment given will not be more than 15 seconds**. This is both to mimic a realistic user scenario and to minimize the query time and the data setup process of the pipeline, which both take enough time as is.
3. **All input videos will have been recorded with the same camera** (and by extension microphone) or screen recordings of the same screen with the same resolution and aspect ratio. This is to minimize the variability of the inputs (except the segments they give, of course).

All of these assumptions are made in service of making the videos as simple to recognize as possible. The aim is to make the only difference in the pipelines be in speed.

Invariances of the system

1. Just like in the lab video query system, the pipeline aims to be *scene and shot invariant*. It should not matter which scene/shot (or even overlap of the two) from a video a user provides, the system should still be able to find it.
2. The pipeline should be able to work for any of the features it is performed on (color histograms, mfccs, audio powers, temporal differences). It should also work for any combination of the four features.

Analysis of the code

The code has been organized in different files with specific purposes:

The `Database.py` file creates and loads the database, which consists of full and cropped videos. All signatures are stored as `.txt` files containing numpy arrays extracted into a dictionary at runtime.

The `Signatures.py` file contains all the methods for computing signatures, which, as explained above, play an important role in our retrieval system. Currently, the system has the capacity to create signatures for four basic features (taken from the lab): `colorhists, mfccs, audio_powers, temporal_diff`. The system can be extended for any combination of those features, including multi-modal ones, simply by running the query with more features inside of it.

For the computation of the score per each video segment in the database compared to the input, the team has created several normalization and evaluation methods in the `Scorer.py` file. The system computes a normalized score for each feature provided to the pipeline and takes the mean score as the final output. A lower score means a better match.

To run the actual pipeline, `VideoQuery.py` is used. This is the most important part of our codebase since it has the methods for the common brute force pipeline and our experimental signature pipeline. In the case of the brute force pipeline, a sliding window is performed over each video in the pipeline which is computed every time, after which the estimated segments and matches are output. For the signature pipeline, the query video's signature is computed and put against all other signatures in the database, outputting a k-best matching list (we have chosen $k = 5$). The videos from that list then get put through the brute-force pipeline as well with the aim of finding a better time estimation for the query and higher accuracy.

Miscellaneous

- The team has decided to use the Librosa library for the MFCC computation and Scikit for the reading of `.wav` files. As this is not the point of the innovation of the project, it should not affect result
- The team used the MoviePy library for the conversion of `.mp4` to `.wav` files.

Evaluation

Evaluation Data

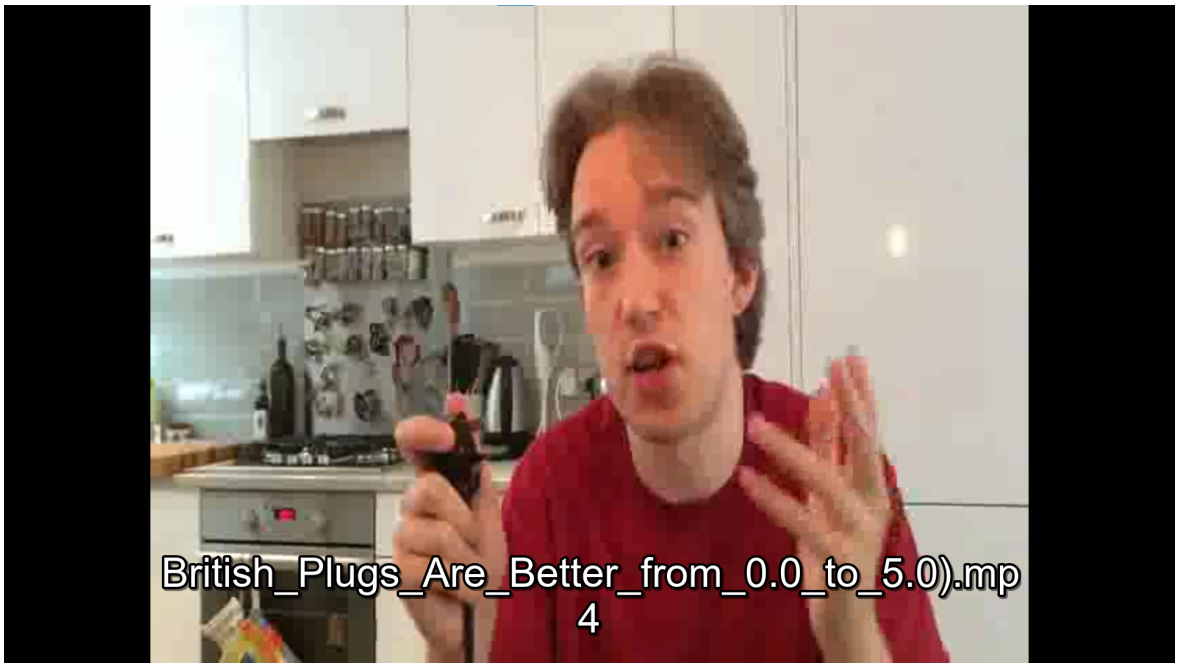
The signature innovation is not related to the localization step, since the computations are entirely separate. Thus, as said in a previous section, perfect localization is assumed. For input data for the query, the team has prepared various well-localized input videos. This includes several screen-recorded and phone-recorded videos, where the majority of the screen is taken up by the input video. This should mimic the effects of perfect localization, while making it still applicable to real-world scenarios (due to the camera-recorded videos-in-videos). As far as their length, to test the system more extensively, the team prepared 5-second segments, 10-second segments and intervals between 5-second segments (e.g.00:07 - 00:12). This way, the weaknesses of the pipeline will have been tested as well.

Results

We decided to use three manually recorded videos to test our system. The results of these are shown below:

Note: The team has decided to focus primarily on `colorhists` and `mfccs` queries, since both team members deemed them the most informative during Lab 5. To illustrate an `audio_powers` and `temporal_diff` example, we have left it at the end of this section, but we will not focus on them.

1. `British_Plugs_Are_Better_from_0.0_to_5.0`



   (a) **Brute-Force Pipeline Results**:

      i. `mfccs`:

```
============================================================
 Query Item ['mfccs'] : ./videos_cropped/British_Plugs_Are_Better_from_0.0_to_5.0).mp4
============================================================

British_Plugs_Are_Better.mp4: 0.0-5.0 0.32635626
DeltaIV.mp4: 125.0-130.0 1317.3785
How_YouTube_Stabilization_Works.mp4: 202.5-207.5 1390.1473
Lemmings.mp4: 0.0-5.0 1402.5886
Your_GPS_Shuts_Down_If_It_Goes_Too_Fast.mp4: 82.5-87.5 1403.1755

============================================================
 Time taken: 31.710352420806885 seconds
```

ii. `colorhists`:

```
============================================================
 Query Item ['colorhists'] : ./videos_cropped/British_Plugs_Are_Better_from_0.0_to_5.0).mp4
============================================================

British_Plugs_Are_Better.mp4: 0.0-5.0 25375.915752998728
The_Artic_Winter_Games_Citation_Needed.mp4: 0.0-5.0 38773.880464641545
Lets_Talk_About_Anstares_Launch_Failure.mp4: 0.0-5.0 40562.65069253954
video_4.mp4: 106.17283950617285-111.17283950617283 41215.611944947916
TUDelft_Electrical_Engineering_Mathematics_Computer_Science.mp4: 0.0-5.0 41251.508982436026

============================================================
 Time taken: 4260.072233200073 seconds
```

(b) **Signature Pipeline Results:**

i. `mfccs`:

```
============================================================
 Query Item ['mfccs'] : ./videos_cropped/British_Plugs_Are_Better_from_0.0_to_5.0).mp4
============================================================

British_Plugs_Are_Better.mp4: 0.0-5.0 0.32635626
DeltaIV.mp4: 125.0-130.0 1317.3785
Lemmings.mp4: 0.0-5.0 1402.5886
video_4.mp4: 9.876543209876544-14.876543209876544 1442.9579
TUDelft_Ambulance_Drone.mp4: 5.0-10.0 1454.1667

============================================================
 Time taken: 24.36710786819458 seconds
```

ii. `colorhists`:

```
============================================================
 Query Item ['colorhists'] : ./videos_cropped/British_Plugs_Are_Better_from_0.0_to_5.0).mp4
============================================================

British_Plugs_Are_Better.mp4: 0.0-5.0 25375.915752998728
The_Artic_Winter_Games_Citation_Needed.mp4: 0.0-5.0 38773.880464641545
Lets_Talk_About_Anstares_Launch_Failure.mp4: 0.0-5.0 40562.65069253954
TUDelft_Electrical_Engineering_Mathematics_Computer_Science.mp4: 0.0-5.0 41251.508982436026
The_Diner_Where_You_Microwave_Your_Own_Food.mp4: 17.5-22.5 42983.83770130763

============================================================
 Time taken: 706.3046703338623 seconds
```

2. `TUDelft_DataScience_Martha_Larson_from_20.0_to_31.0`

(a) **Brute-Force Pipeline Results**:

i. `mfccs`:

```
============================================================
 Query Item ['mfccs'] : ./input-videos/TUDelft_DataScience_Martha_Larson_from_20.0_to_31.0.mp4
============================================================

TUDelft_Ambulance_Drone.mp4: 52.2-63.853899999999996 1257.6495
DeltaIV.mp4: 5.8-17.453899999999997 1396.7142
Asteroid_Discovery.mp4: 0.0-11.6539 1478.5515
supercoilsNL.mp4: 5.8-17.453899999999997 1680.5547
supercoilsEN.mp4: 5.8-17.453899999999997 1737.5162

============================================================
 Time taken: 98.51005172729492 seconds
```

ii. `colorhists`:

(b) **Signature Pipeline Results:**

i. `mfccs`:

```
============================================================
 Query Item ['mfccs'] : ./input-videos/TUDelft_DataScience_Martha_Larson_from_20.0_to_31.0.mp4
============================================================

supercoilsEN.mp4: 5.8-17.453899999999997 1737.5162
video_07.mp4: 5.805805805805806-17.459705805805807 1946.2034
TUDelft_cheap_solar_cell.mp4: 98.6-110.2539 2747.0334
TUDelft_animation_efficient_and_sustainable.mp4: 87.0-98.65390000000001 2774.248
Lemmings.mp4: 0.0-11.6539 2813.337

============================================================
 Time taken: 84.21807837486267 seconds
```

ii. `colorhists`:

```
============================================================
 Query Item ['colorhists'] : ./input-videos/TUDelft_DataScience_Martha_Larson_from_20.0_to_31.0.mp4
============================================================

Your_GPS_Shuts_Down_If_It_Goes_Too_Fast.mp4: 0.0-11.6539 263194.6291467285
TUDelft_Ambulance_Drone.mp4: 81.2-92.85390000000001 264940.4877141679
Google_Street_View_Race.mp4: 17.4-29.0539 266583.50989708444
DeltaIV.mp4: 63.8-75.4539 267627.3306313036
Danger_Humans.mp4: 17.4-29.0539 273887.26921217033

============================================================
 Time taken: 1093.6275804042816 seconds
```

3. `Delta|V_from_13.0_to_17.0.mp4`

(a) **Brute-Force Pipeline Results:**

i. `mfccs:`

```
============================================================
 Query Item ['mfccs'] : ./input-videos/DeltaIV_from_13.0_to_17.0.mp4
============================================================

DeltaIV.mp4: 20.266666666666666-25.366666666666667 764.56635
Asteroid_Discovery.mp4: 0.0-5.1 785.8697
Your_GPS_Shuts_Down_If_It_Goes_Too_Fast.mp4: 83.6-88.7 789.04755
TUDelft_Ambulance_Drone.mp4: 40.53333333333333-45.63333333333333 828.5912
How_YouTube_Stabilization_Works.mp4: 228.0-233.1 872.67303

============================================================
 Time taken: 67.73005723953247 seconds
```

ii. `colorhists:`

```
============================================================
 Query Item ['colorhists'] : ./input-videos/DeltaIV_from_13.0_to_17.0.mp4
============================================================

Asteroid_Discovery.mp4: 12.666666666666666-17.766666666666666 179025.5757047052
Welcome_To_Life.mp4: 17.733333333333334-22.833333333333332 211649.3672494924
Kerbal_Space_Program1.mp4: 0.0-5.1 216294.3700503075
Kerbal_Space_Program2.mp4: 0.0-5.1 217356.3709475717
supercoilsNL.mp4: 0.0-5.1 231818.23591408585

============================================================
 Time taken: 26635.829841852188 seconds
```

(b) **Signature Pipeline Results:**

i. `mfccs`:

```
============================================================
 Query Item ['mfccs'] : ./input-videos/DeltaIV_from_13.0_to_17.0.mp4
============================================================

DeltaIV.mp4: 20.266666666666666-25.366666666666667 764.56635
Asteroid_Discovery.mp4: 0.0-5.1 785.8697
Your_GPS_Shuts_Down_If_It_Goes_Too_Fast.mp4: 83.6-88.7 789.04755
TUDelft_Ambulance_Drone.mp4: 40.53333333333333-45.63333333333333 828.5912
How_YouTube_Stabilization_Works.mp4: 228.0-233.1 872.67303

============================================================
 Time taken: 21.407542943954468 seconds
```

ii. `colorhists`:

```
============================================================
 Query Item ['colorhists'] : ./input-videos/DeltaIV_from_13.0_to_17.0.mp4
============================================================

Asteroid_Discovery.mp4: 12.666666666666666-17.766666666666666 179025.5757047052
Kerbal_Space_Program1.mp4: 0.0-5.1 216294.3700503075
Kerbal_Space_Program2.mp4: 0.0-5.1 217356.3709475717
Why_Does_Nighttime_Smartphone_Footage_Look_All_Flickery.mp4: 25.333333333333332-30.433333333333334 276115.2873256801
Lemmings.mp4: 12.666666666666666-17.766666666666666 282781.8140926387

============================================================
 Time taken: 1178.0245678424835 seconds
```

Demonstration of `audio_powers` and `temporal_diff`

1. **Brute-Force Pipeline Results:**

(a) `audio_powers`:

```
============================================================
 Query Item ['audio_powers'] : ./input-videos/DeltaIV_from_13.0_to_17.0.mp4
============================================================

DeltaIV.mp4: 98.8-103.9 0.017102899342982848
Welcome_To_Life.mp4: 146.93333333333334-152.03333333333333 0.019710942609033385
Civile_Techniek1.mp4: 32.93333333333333-38.03333333333333 0.02372460117425388
video_07.mp4: 15.215215215215215-20.315215215215215 0.023836305403077972
TUDelft_Ambulance_Drone.mp4: 126.66666666666667-131.76666666666668 0.025515863091438214

============================================================
 Time taken: 41.17835593223572 seconds
```

(b) `temporal_diff`:

```
============================================================
 Query Item ['colorhists'] : ./input-videos/DeltaIV_from_13.0_to_17.0.mp4
============================================================

Asteroid_Discovery.mp4: 12.666666666666666-17.766666666666666 179025.5757047052
Welcome_To_Life.mp4: 17.733333333333334-22.833333333333332 211649.3672494924
Kerbal_Space_Program1.mp4: 0.0-5.1 216294.3700503075
Kerbal_Space_Program2.mp4: 0.0-5.1 217356.3709475717
supercoilsNL.mp4: 0.0-5.1 231818.23591408585

============================================================
 Time taken: 26635.829841852188 seconds
```

2. **Signature Pipeline Results:**

   (a) `audio_powers`:

```
============================================================
 Query Item ['audio_powers'] : ./input-videos/DeltaIV_from_13.0_to_17.0.mp4
============================================================

DeltaIV.mp4: 98.8-103.9 0.017102899342982848
Welcome_To_Life.mp4: 146.93333333333334-152.03333333333333 0.019710942609033385
Civile_Techniek1.mp4: 32.93333333333333-38.03333333333333 0.02372460117425388
video_07.mp4: 15.215215215215215-20.315215215215215 0.023836305403077972
TUDelft_Ambulance_Drone.mp4: 126.66666666666667-131.76666666666668 0.025515863091438214

============================================================
 Time taken: 41.17835593223572 seconds
```

   (b) `temporal_diff`:

```
============================================================
 Query Item ['temporal_diff'] : ./input-videos/DeltaIV_from_13.0_to_17.0.mp4
============================================================

TUDelft_DataScience_Hayley_Hung.mp4: 124.13333333333334-129.23333333333332 92513174.89779757
How_YouTube_Stabilization_Works.mp4: 91.2-96.3 106341901.92437518
British_Plugs_Are_Better.mp4: 7.6-12.7 137605969.09291527
video_4.mp4: 154.68802135468803-159.78802135468803 139636879.77084222
Google_Street_View_Race.mp4: 25.333333333333332-30.433333333333334 142595085.1338346

============================================================
 Time taken: 1277.0308248996735 seconds
```

Demonstration of Multimodal Signature Query - MFCC and Color Histograms

```
============================================================
 Query Item ['mfccs', 'colorhists'] : ./videos_cropped/British_Plugs_Are_Better_from_0.0_to_5.0).mp4
============================================================

British_Plugs_Are_Better.mp4: 0.0-5.0 12688.121054630348
The_Artic_Winter_Games_Citation_Needed.mp4: 0.0-5.0 20229.86814980124
Lets_Talk_About_Anstares_Launch_Failure.mp4: 0.0-5.0 21162.89175251977
TUDelft_Electrical_Engineering_Mathematics_Computer_Science.mp4: 0.0-5.0 21480.146825202388
The_Diner_Where_You_Microwave_Your_Own_Food.mp4: 17.5-22.5 22293.07992243116

============================================================
 Time taken: 1154.9196140766144 seconds
```

Conclusions from the data

The results from the experimental queries showcase the potential of our innovation and how the assumptions of the experiment got in their way.

Speedup from the signatures is always noticeable, both for visual and audio features, but much more for the former. Visual features, due to being more computationally intensive seem to benefit far more from the pre-computed signatures, often outputting a result more than twice as fast. Audio features on the other hand still have a speedup, but nowhere near as substantial. This most likely attributes itself to the fact that they work in fewer dimensions and that audio computations are faster in general.

Where the pipeline fails, however, is in the actual accuracy of each query. The assumption of perfect localization did not seem to hold up in most situations as much as was expected. Instead, the only time the queries got optimal results was when querying actual database videos, The query (even the brute-force version) sometimes fails to find even the exact matches of the video given to it. However, the cases where the signature pipeline failed seemed to coincide with the times the brute force pipeline failed as well.

This means the accuracy issues are most likely due to improper assumptions about the input data, rather than any substaintial difference between the two pipelines. In the case the brute-force pipeline found the correct match in the top-k results, it was also found by the signature pipeline. The speed improvements also held up in every run. Thus, once the pipeline assumptions are cleared up and combining the signatures with a proper localization pipeline should yield even better results.

Issues Encountered During the Project

The team faced lots of challenges in the process, mainly in data pre-processing and the actual brute-force pipeline. Running an input video on such a huge database, combined with the intensity of each computation limited the number of iterations and testing possible. At many points, the team was not sure whether problems in the output were the result of hard query videos or implementation issues. The way the team tried to remedy was by runnning the pipeline on input videos from the actual database.

Another issue encountered was the normalization of the data. Some features have different scales and cannot be directly compared with each other, and thus they will be weighted differently. The way the team tried to remedy this was through normalization functions.

Future Improvements

The system has a lot of room for growth. Were it not for the time constraints of the experiment, more would have been done and improved. Currently, setting up the database takes up a lot of effort and the overall result is still slow. Additionally, there is an undeniable loss in the accuracy, as seen by the evaluation results.

A way to improve the accuracy would be building a tree-like structure for each signature. Currently, the system only has a signature for each 5-second segment (where 5 is in the end just an arbitrary number that was thought reasonable). What about the queries between those segments? As was seen in the evaluation results, the system has a harder time recognizing videos for videos spanning multiple signatures. The way the system can remedy this could be adding more segments, e.g. for each 10-seconds, or 20-seconds until it reaches the duration of the entire video. This tree would also be able to grow downward for higher accuracy, creating 2.5-second, 1.25-second and shorter segments. In theory, this should improve the accuracy of the pipeline even more by sacrificing a bit of speed. However, due to the time constraints of the project, this has not been tested yet.

Another possible improvement would be more efficient storing of the data. Currently, all signatures are stored as NumPy arrays in `.txt`. This makes the size of the system larger than desired, an issue that might get worse by adding more data and/or building a tree as mentioned in the previous improvement. If the data could be directly stored as bytes, it would likely take up far less space.

Conclusion

The system manages to achieve a substantial computational improvement despite the setbacks in accuracy. Hindrances from extensive assumptions aside, the queries are faster and still accurate (when given optimum conditions), thus confirming that there is ground for the initial hypothesis. With future improvements and combining it with an actual localization pipeline, this innovation may end up practically useful in many settings similar to Video Shazam.