Flexible Motion In-betweening with Diffusion Models

Aditya Kumar

Introduction

Motion in-betweening is the task of generating motion sequences to interpolate user-provided keyframes constraints. The paper uses diffusion models to generate diverse human motions based on text as well as keyframe constraints. The authors have named this model **Conditional Motion Diffusion In-betweening (CondMDI)**.

Motivation

- The results that are generated by the model have minor footskates and jitters for highly dynamic motions.
- One of the reasons for these footskates and jitters are the outliers in the dataset
- The authors have identified that the HumanML3D dataset used to train the model contains skating and swimming data which is the reason for the footskates and decreased smoothness in the model.
- Our implementation removes these outliers by sampling the dataset based on the similarity of the data with **swimming** and **skating** text prompts.

Goal

- To remove the outliers from the dataset and see if the results improve in terms of footskates and smoothness.
- To decrease the trajectory error and improve the R_precision of the model as compared to the previous model.

Results

Results before the gaussian sampling

```
---> [ground truth] R_precision: (top 1): 0.5147 (top 2): 0.7116 (top 3): 0.8068
---> [vald] Trajectory Error: (traj_fail_20cm): 0.9922 (traj_fail_50cm): 0.4912 (kps_fail_20c
m): 0.9166 (kps_fail_50cm): 0.3049 (kps_mean_err(m)): 0.6638
---> [vald] Keyframe Error: 0.7152
---> [vald] Skating Ratio: 0.1000
---> [vald] Matching Score: 9.7807
[vald] R_precision: (top 1): 0.0332 (top 2): 0.0566 (top 3): 0.0791
```

Results after the gaussian sampling

```
---> [ground truth] R_precision: (top 1): 0.5147 (top 2): 0.7116 (top 3): 0.8068
Notebook editor cells
rajectory Error: (traj_fail_20cm): 0.9912 (traj_fail_50cm): 0.4922 (kps_fail_20c
m): 0.9184 (kps_fail_50cm): 0.3055 (kps_mean_err(m)): 0.6677
---> [vald] Keyframe Error: 0.7187
---> [vald] Skating Ratio: 0.1000
---> [vald] Matching Score: 9.8124
---> [vald] R_precision: (top 1): 0.0312 (top 2): 0.0742 (top 3): 0.1162
Time: 2024-11-03 19:52:31.495047
```

Observation

- The **Trajectory Error** for 20cm has decreased but for remaining it has increased.
- The **Keyframe Error** increases from 0.7152 to 0.7187
- The **Skating Score** remains the same
- **Matching Score** increases from 9.7807 to 9.8124
- **R_precision** value for top 1 decreases slightly but there is considerable improvement in top 2 and top 3 **R_precision** values.

Conclusion

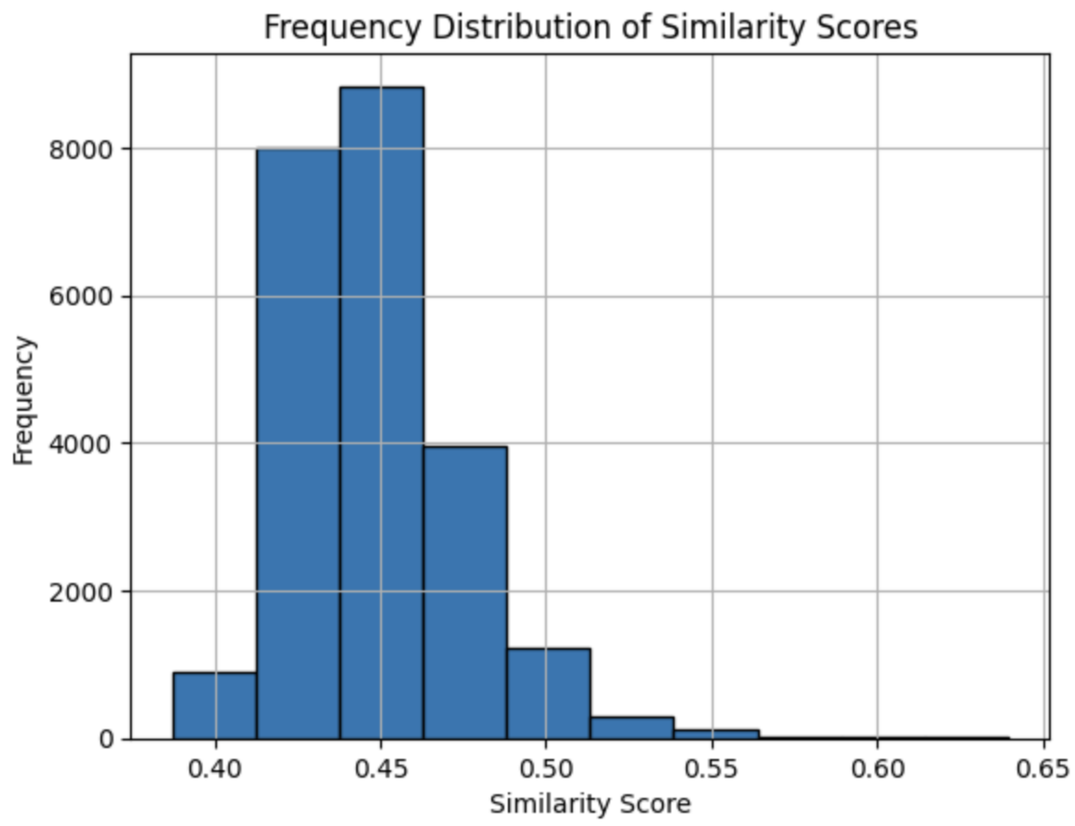
The **R-Precision** and the **Matching Score** have improved as expected.
The unexpected rise in **Trajectory error** and **Keyframe error** can have 2 plausible explanations:

- The model was trained for only 5 epochs as comapred to 120000 epochs in the original paper. This could have led to the model not learning the correct motion sequences.
- We have decreased the number of keyframes in the dataset to 0.9 times the original dataset (i.e. 13181 keyframes instead of 14646 keyframes). This could have led to the model not learning the correct motion sequences as the keyframes are the only source of information for the model to learn the motion sequences.

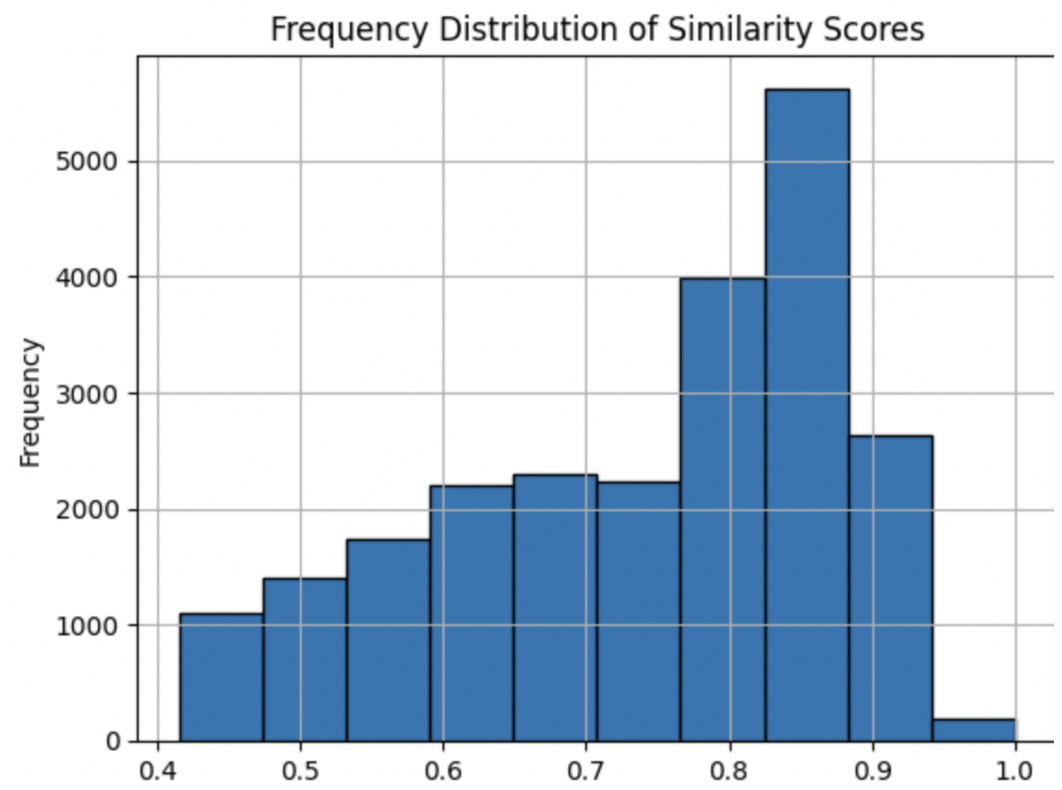
Method

My method for removing the outliers from the dataset can be divided into 2 parts:

- **Scoring the data -** Before we can sample the data we first will have to score them based on some metrics do that we obtain an ordering in the dataset. For scoring the dataset we tried both **bag-of-words** and **word2vec** approaches to compute the similarity of text promts with the words **swimming** and **skating**. We found that the **word2vec** approach gave a much better distribution of scores that approximated a Gaussian distribution as compared to the **bag-of-words** approach. We even applied the **TFIDF** algorithm to give appropriate weights to the most improtant words when finding the similarity score. Below is the plot of the scores we obtained using the **word2vec** approach.



To compare here is the same plot for the scores obtained using the **bag-of-words** approach.



- **Sampling the data -** Based on the distribution of score obtained above we sampled the data using a Gaussian sampling.
  Let $Z(\boldsymbol{X}) =$ similarity score of the promt $\mathbf{X}$ obtained above.
  Let μ and σ be the mean and covariance matrix of the scores of the dataset.
  Then the probability of sampling $\mathbf{X}$ is given by the Gaussian distribution as:

$$p(\mathbf{X}|\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(Z(\boldsymbol{X})-\mu)^2}{2\sigma^2}}$$

  We sampled 90% of the data using this Gaussian distribution and trained the model on this new dataset.