Text Query Video Retrieval at Scene Granularity

Corey Xing* corey.xing@gatech.edu
Daanish Mohammed* dmohammed7@gatech.edu
Junha Lee* jlee3551@gatech.edu
Shez Malik* smalik46@gatech.edu
*Georgia Institute of Technology

Abstract

We present a method for Text Query based Video Retrieval. A video generally consists of a composition of several scenes. Our proposed method will allow a user to enter a text query, and retrieve the scene from the video that is most accurately described by the entered query. Our method consists of two main components: (i) Scene Segmentation, where a video is segmented into different scenes, and (ii) Video and Text Matching, where each scene in the video as well as the text query are encoded into a shared feature space, and the cosine similarity is used to determine the scene that best matches the text query. Finally, the model ranks all the possible scene segments in order of highest similarity score to lowest similarity score. After both components of the model were working, it was tuned and evaluated on validation data. The model selected the correct scene in the top 1% of possible results 37.34% of the time, 85.32% of the time in the top 10% of results, and 95.77% of the time was the correct scene in the top 25% of the predicted results. From end-to-end, the system will produce the correct scene segment from a text query in the top-10% of results $\mathbf{36.2}\%$ *of the time.*

1. Introduction

Videos have conquered the internet. In 2022 alone, videos will account for 80% of the total global internet traffic [2]. Consequently, with the popularization of video sharing, it becomes nearly impossible for people to get desired information from such an expansive data space. As a result, the demand for efficiency and convenience from intuitive search engines has become undeniably pertinent. A ”shot” is defined as a segment of video frames that are captured by the same camera over an uninterrupted time frame, and a ”scene” is defined as a group of shots that depict a significant detail to a story [13]. This implies that scenes specifically are valuable storytelling mechanisms within movies. Therefore, we aim to produce a system that can achieve video retrieval at scene granularity as this would play a vital role in enabling end users to efficiently search for the most relevant frames within a video through a text query. This is valuable as it not only helps users to quickly find the desired scene as opposed to searching through the entirety of a given video but it also provides the search algorithm with a more efficient mechanism to parse a received video as many alternatives suggest analyzing each frame of the video or by using a fixed sliding window [1] [10] [5] [7] [11] [4] neither of which benefit from producing a valuable storytelling mechanism such as a scene nor receive the optimization of having less data to analyze from a given video.

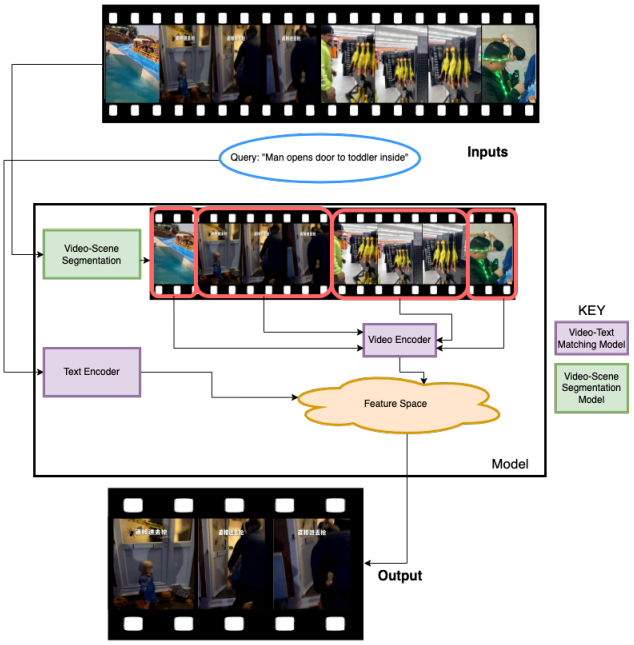


Figure 1. Model architecture operating on an example YouTube video.

We propose a system (Fig 1.) that takes two inputs: an arbitrary video and a text query from the user. The sys-

tem will then intelligently segment the video into different scenes while simultaneously passing the text input through a text encoder. The system will then pass the clips representing scene segments of the initial video into the video encoder system, which, transforms these keyframes into the same feature space as the output of the text encoder. From there, the model outputs the final video segment at scene granularity.

2. Related Works

After a thorough review of the existing literature, to the best of our knowledge, there is no existing work that achieves video scene retrieval from a given text query. Prior work in the field can be broken down into 3 distinct categories: Text query video retrieval from a corpus of videos, temporal localization within a single video, and language-based video retrieval.

2.1. Temporal localization within a single video

The majority of the strategies, apply a sliding window with fixed size and approach the problem of classification as a regression problem[sources for sliding window] or they split the video into multiple segments in advance [8]. Specifically looking at the works that leverage pre-segmentation [8] [7] [17], Hendrick et al. aim to acquire the temporal relationship between subsequent events within a clip [8] [7]. They accomplish this through moment localization which treats video context as a latent variable and unifies prior moments in the clips. They do this through several pre-segments that are referred to as 'contexts' and leveraging prior context and subsequent context to the retrieved base context to determine if the base context matches the proposed query. Therefore, the pre-segmentation occurs agnostic to specific scenes. This is beneficial when trying to find correlated temporal events within a singular video, but if a user was watching a basketball highlight video and wanted to see "a three pointer" this model, along with the others attempting to acquire temporal context of certain events would not fair well as it would not be able to acquire the frames specific to the scene that the user desires.

2.2. Language based retrieval

Moreover, another approach to query-based video retrieval is language-based retrieval, which aims to enhance temporal localization by attempting to optimize the search mechanism [10] [4] [8] [1] [5] [9]. This is achieved by grounding natural sentences in untrimmed videos. Temporal GroundNet(TGN) produced by Chen et al. aims to leverage fine-grained interactions between video frames and words in a natural sentence [1]. For example, by considering an arbitrary video and query: "woman flies kite" TGN analyzes an untrimmed video frame by frame without considering temporal locality and searches for what it understands to be a "woman" and if that is found near "flying" and "kite" then output the frames that are closest to each other and are relevant to the query. While again a valuable tool, these networks that leverage grounding sentences within videos are limited because they fail to produce an output relevant to the granularity of a single scene.

2.3. Text query video retrieval from a corpus of videos

Video retrieval from a corpus of videos is essential for large media platforms like YouTube, TikTok, Vimeo, and the like. Prior work in this area [11] [4] typically returns an output of a relevant whole video from a large database of possible matches either via concept matches(matching visual or semantic concepts) or common feature space methods(feature extraction and mapping to a common space where the matching can be better optimized via metric learning) [3]. However, both of these fail to achieve the goal of producing the output of a specific segment of a video as their primary concentration is to find the match within a corpus of videos as opposed to a specific input video. Sun et al. attempt to not only produce this match within a vast database [14], but also return a specific segment of the video all in a singular network. In order to accomplish this, the paper leverages a text-aligned attention mechanism similar to what is leveraged in language-based segment retrieval to efficiently generate temporal context and a ranking strategy that aims to improve the performance of video corpus segment retrieval. Again, however, the primary limitation is that they fail to consider the value of returning a storytelling mechanism as powerful as a specific scene.

3. Technical Approach

3.1. Scene Segmentation

3.1.1 Overview

Before we dive into an explanation of the Scene Segmentation model used, it's important to first understand the problem at hand. The existing literature makes a distinction between the terms "shots" and "scenes". In general, a shot is a sequence of frames that are captured by a camera for an uninterrupted period of time, and therefore it is visually continuous. On the other hand, a scene is a composition of shots used to tell a part of the story, which makes it a semantic unit at a higher level.

Scene segmentation can be viewed as a binary classification problem. Given a sequence of shots, the objective is to determine what shot boundaries constitute a scene boundary. In other words, we assign each shot boundary a label $\in \{0, 1\}$, "0" implying that the shot boundary represents a within-scene transition, and "1" implying that the shot boundary represents a cross-scene transition.

Our model architecture is heavily influenced by Rao et al. [12]. The model first passes each movie through a Shot Detection module to split the movie into the sequence of shots that constitute the movie. Each shot is then passed through a Holistic Feature Extraction module to extract high-level features from key frames that make up each shot. The authors of [12] use a set of 4 features: place, cast, action, audio, which incorporates multi-modal data. However, in our model we only used the place features, obtained by passing key frames that make up each shot through a ResNet50 [6] model that was pre-trained on the Places dataset [18] for the task of scene recognition (note that "scene recognition" in this context refers to the recognition of the background or scene in an image). These shot features are represented as $[s_1, ..., s_n]$. Next, a Boundary Network (BNet) made up of temporal convolutional layers takes a clip of the movie containing $2w_b$ shots as input and returns a representation of the shot boundary b_i. This step of modeling the shot boundary is represented by (1).

$$b_i = BNet([s_{i-(w_b-1}, ..., s_{i+w_b}) \tag{1}$$

After we get the representations of the shot boundaries b_i, the next stage of the pipeline uses a segment-level model to predict coarse scores for the scene boundaries $\bar{o_i}$. At first, a Bi-LSTM takes in a clip of w_t shots and it outputs the variables p_i, which represent the probability that the i'th shot boundary is a scene boundary, and then a hard classification is performed to get the course predictions $\bar{o_i}$. This step of making coarse prediction at the segment level is represented by (2a) and (2b).

$$p_i = BiLSTM([b_1, ..., b_{n-1}]) \tag{2a}$$

$$\bar{o_i} = \begin{cases} 1 & p_i > \tau \\ 0 & \text{otherwise} \end{cases} \tag{2b}$$

The coarse scores produced by the segment-level model are not the best, since it only considers local context over w_t shots, while ignoring global context over the entire input video/movie. The authors of [12] take the shot representations s_i as well as the coarse scene boundary predictions $\bar{o_i}$, and pass it to a Global Optimal Grouping model to output the predicted scene boundaries o_i. This step is represented by (3).

$$[o_1, ..., o_{n-1}] = Grouping([s_1, ..., s_n], [\bar{o_1}, ..., \bar{o_{n-1}}]) \tag{3}$$

The overall Scene Segmentation model architecture is shown in Fig 2.

3.1.2 Training

The scene segmentation model was trained on a subset of the MovieNet dataset (more information in Section 3.3).

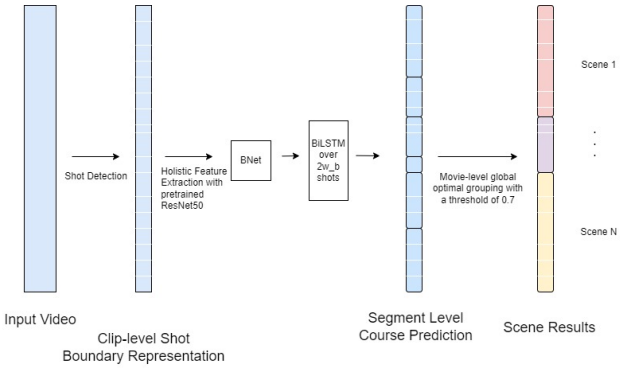


Figure 2. Scene Segmentation Model Architecture.

Since it is a binary classification problem, the loss function used is the cross entropy loss. Since there was an imbalance in the class labels (0's occurred about 9 times as much as 1's), we assigned relative weights of 1 and 9 for the labels 0 and 1 respectively in the cross entropy loss function, similar to Rao et al. [12]. The model was trained for 30 epochs using the Adam optimizer, with the initial learning rate set to 0.01, and then reduced to 0.001 after 15 epochs. We used a batch size of 8. The configuration of the system used to train this model is described in section 3.3.

3.2. Video-Text Match

3.2.1 Overview

Our architecture involves a text encoder and a video encoder that will transform the given text/video to the same feature space, after which we compute the cosine similarity between the feature vectors as the matching score. The task of transforming a sentence query into the same feature space as the video frames is significantly reliant on the sequence of the words; therefore, for the text encoder, we have used a standard LSTM as it is a sequence-based model. For the video encoder, we use a CNN-LSTM architecture, where a video (as a sequence of frames) is passed through a CNN, which extracts spatial features from the individual frames, and the output sequence is passed into an LSTM in order to extract the temporal information from the sequence. Due to the computational cost of passing an entire stack of frames through a CNN, our selected CNN encoder architecture based on both performance and computational complexity. We settled on using RegNetY800MF [16] which has only 6.4 million parameters but achieves comparable accuracy on ImageNet to many other networks several times the size, such as ResNeXt50 [15]. To help speed up and simplify training, we initialized the CNN with pre-trained weights. For purposes of simplicity, our architecture takes into account only the image data from the video, excluding the audio data; however, there is a potential extension for our work in the future, which is later discussed in section

6.1.

At inference time, a video is first passed through our scene segmentation model to get scene boundary labels. Then, each scene can be processed separately by the video encoder contained within the text-video matching model. This creates a feature representation of each scene in the video that can then be stored for later reference. This step is the most expensive aspect of inference but only needs to be done once per video. When provided a text query from a user, we then pass this query through the text encoder module of the model and compute the similarity scores against each of the already processed scenes. Finally, the output sorts the scenes by similarity to find the best matches. This step is relatively inexpensive as the scene representations have already been computed, so only the text input needs to be passed through the model. Thus, querying the model can be done in real-time.

The overall architecture is depicted in Figure 3.

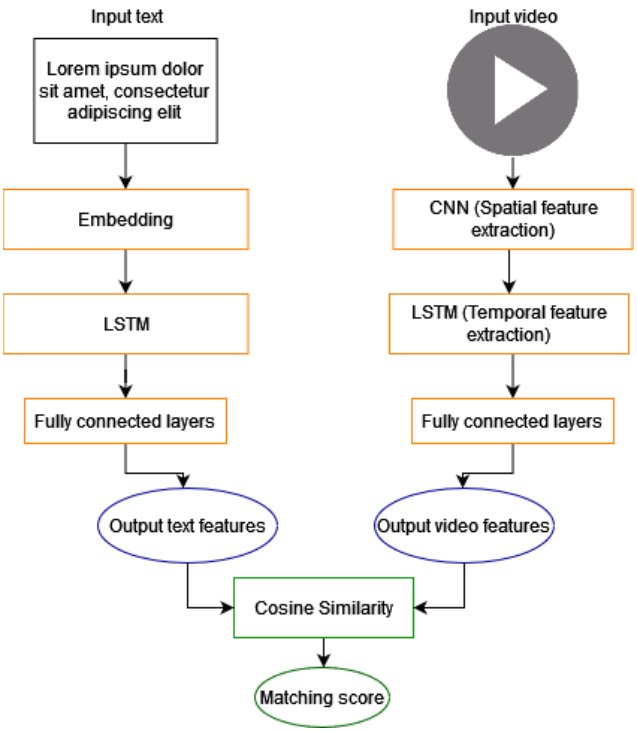


Figure 3. Scene Segmentation Model Architecture.

3.2.2 Training

The loss criterion used to train the video-text matching model is as follows: for a specific sample k consisting of a video and a caption, $L = \frac{CE(S_{tk},T_k)+CE(S_{vk},T_k)}{2}$, where S_{tk} is the similarity between the specific sample caption against all videos in the batch, S_{vk} represents the similarity between the specific sample video against all captions in the batch, and T_k is mean of the similarity between the sample video and all videos in the batch, and the similarity between the sample caption and all captions in the batch. Additionally, CE denotes cross-entropy loss. Thus, this criterion results in the model learning to maximize the similarity between a video v_j and a caption t_i if the corresponding videos v_i and v_j are similar, or the corresponding text t_i or t_j are similar, and minimizing it in the opposite case. During training, many compromises had to be made to account for deficiencies caused by hardware constraints (details in section 3.3) in both GPU memory and speed. Training clips were first downsampled to 2 fps, down from their source frame rate of 30; higher frame rates would make it impossible to for the forward and backward pass to fit in GPU memory, despite our economical choice of CNN. This same restriction also impacted our choice of batch size, which we found could be set no higher than 4. Considering that our loss function directly relies on comparing samples to other samples in the same batch, a fairly large batch size would be appropriate. To help compensate for this, we chose to accumulate gradients over multiple batches before performing our update step rather than doing so every iteration. We ultimately selected an accumulation interval of 32. The cost of training also prevented us from being able to utilize the entire training set. Our best performance was achieved using 4 captions per video, which took roughly 25 hours to train over 10 epochs. After experimentation, a learning rate of .0001 was used. Further details on experimentation are in section 4.

3.3. Technical Details

3.3.1 Datasets

The dataset we utilize in training the Video-Text Matching Model is Vatex, a dataset consisting of Youtube clips, with 26K training clips and 3K validation clips (though only 23K and 2.5K videos could be downloaded successfully). Each clip has 10 associated English language captions, giving us a total of 230K training examples. The Vatex dataset is distributed as a JSON file containing the video IDs and captions.

In training the scene segmentation model we utilized a subset of MovieNet, a dataset comprised of 1,100 movies with a vast amount of multi-modal data. MovieNet includes 1.1M characters with bounding boxes and identities, 42K scene boundaries, 2.5K aligned description sentences, and 92K tags of cinematic style.

3.3.2 Hardware

The scene segmentation model was trained on an NVIDIA Tesla T4 GPU with 16 GB of RAM on Google's Cloud Environment. The video text-matching model was trained on an NVIDIA RTX 3070 GPU with 8 GB of RAM, we were

constrained by both GPU memory and speed. Lastly, the whole system was created using PyTorch 1.12 and Cuda 11.3.

4. Experimentation and Results

Our approach to the problem of Text Query based Video Retrieval at Scene Granularity consists of two main components: Scene Segmentation and Video and Text Matching. We present our results from these two components in 4.1 and 4.2.

4.1. Scene Segmentation

For the scene segmentation component of our pipeline, our work has been highly influenced by Rao et al. [12], who proposed a local-to-global approach to the scene segmentation problem by integrating multi-modal information across three levels (clip, segment and video) in order to learn the temporal structure of scenes as well as the complex semantic information contained in movies. The main difference between our model and Rao et al. [12] is our choice to ignore multi-modal data. While Rao et al. use a set of 4 features place, cast, action, audio in their Holistic Feature Extraction module, we chose to only use the "place" features. As we show in Table 1, our decision to only use the place features did not have a significant negative impact on the performance of the model; in fact it helped simplify the model architecture quite a bit. We also used a different set of hyperparameters (mentioned in section 3.1.2), and we found that a threshold of 0.7 (2b) worked best with our configuration.

Our scene segmentation model was trained on a small subset of the MovieNet dataset for a period of 30 epochs. The training and validation loss curves are shown in Fig 3. The results of our model evaluated on a test set are shown in Table 1, along with the results obtained by by Rao et al's [12] model as well as those obtained by randomly guessing which shot boundaries are actually also scene boundaries.

Model	AP
Our Scene Segmentation Model	**0.424**
Rao et al's Multi-Modal LGSS model	0.471
Random Guessing	0.082

Table 1. Evaluation metrics for our Scene Segmentation model on the test set. (Average Precision (AP) is taken over the scene boundary transition predictions, i.e. when $o_i = 1$)

From the results in Table 1, we see that our decision to exclude the cast, action and audio features, as well as to train on only a subset of the MovieNet dataset did not result in a significant negative impact on the performance. In fact, it helped us avoid a large amount of computation which would have been needed, resulting in significantly longer training times.

(a) Train loss curves

(b) Validation loss curves

Figure 4. Train and Validation loss curves for the Baseline SceneSeg model

4.2. Video and Text Matching

The experimentation for the video-text matching task was conducted leveraging the Vatex dataset (23K training, 2.5K validation) detailed in section 3.3 and the loss criterion defined in section 3.2. A third-party tool was used to download the videos, and a custom loader was written for the dataset. This loader is responsible for producing individual video-caption pairs, loading the videos into memory as tensors, and mapping the individual tokens of the caption to indices. To help prevent overfitting, we also mask out any token that does not appear frequently enough in the training dataset (at least 5 times in total). Lastly, for hyperparameter tuning, we trained the models with an Adam optimizer for 30 epochs or until diminishing returns were noticed. Due to hardware constraints, the time to train the model was incredibly significant (2 hours/epoch) and limited the scope of experimentation to hyperparameter tuning.

For experimentation, three different configurations were tested. The parameters modulated were the learning rates,

Config	Learning Rates	Captions per Video	Output Dimensionality
Config 1	5e-4	1	512
Config 2	1e-4	4	512
Config 3	1e-4	1	512
Config 4	5e-5	1	1024

Table 2. Different hyperparameter configurations used during experimentation

the number of captions used per video, and the final feature space dimensionality for similarity score calculation. These configurations are detailed in Table 2.

Configuration	Top 1% Accuracy	Top 10% Accuracy	Top 25% Accuracy
2	37.34%	85.32%	95.77%
3	22.37%	72.11%	90.36%
4	7.55%	43.93%	74.17%

Table 3. Model accuracy evaluations for the configurations in Table 2. Top 1% Accuracy refers to the percentage at which the model predicted the correct video within the top 1% of results. This logic is expanded to the rest of the columns.

When tuning the learning rate, it proved that higher learning rates caused learning to degenerate early on, which resulted in all inputs being mapped to an identical embedding. When the learning rate was set to 5e-4, the average training loss stabilized at around 0.8 and around 0.91 for the average evaluation loss.

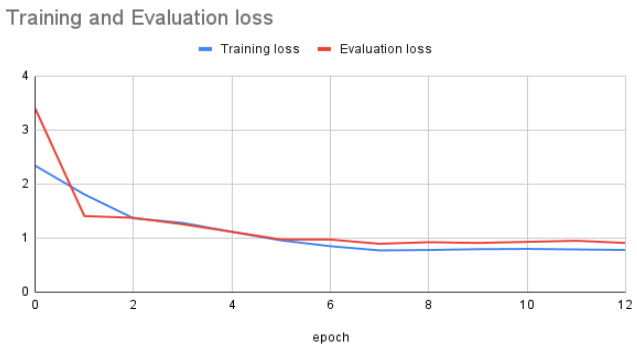
When the learning rate was further reduced to 1e-4, the loss values broke this floor and the model was able to stabilize around 0.18 for average training loss and around 0.4 for average evaluation loss. Lastly, when the learning rate was halved to 5e-5, the model appeared to overfit the training data by producing an average training loss of 0.12, but an average evaluation loss of 0.79 (Figure 5). The evaluation results for the aforementioned trainings and configurations are mentioned in Table 3

Moreover, the experiments were initially conducted by using just 1 caption from each video, but further experimentation found that using an increased number of captions was key to achieving better evaluation performance. Interestingly, changing the dataset size did not seem to impact the speed at which the model converged. This implies that overfitting is likely the limiting factor, and that allowing more captions per video introduces a key form of regularization. Lastly, upon increasing the output feature space to 1024, the model significantly overfits the training data and produced an accuracy of only 7% in the top 1% of predicted videos.

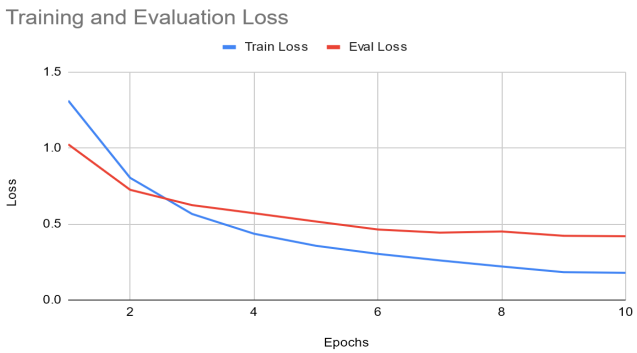
5. Conclusion

5.1. Summary

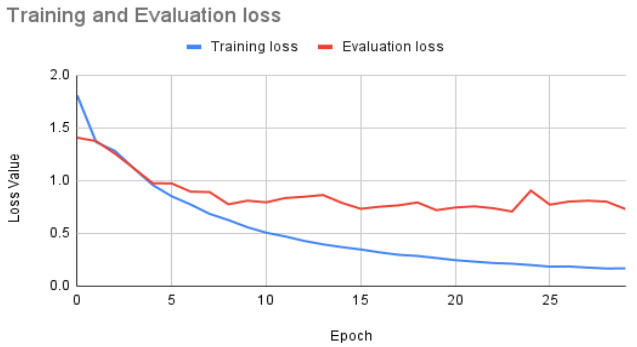
The final goal of creating a model that can intelligently retrieve scenes from a video upon being passed a text query



(a) Training loss and evaluation loss curves when learning rate was set to 5e-4



(b) Training loss and evaluation loss curves when learning rate was set to 1e-4 with 4 captions per video



(c) Training loss and evaluation loss curves when learning rate was set to 5e-5 with 1 caption per video

Figure 5. Training and Validation loss curves for the text-video matching model hyperparameter tuning.

was achieved. The scene segmentation model was able to achieve 0.424 Average Precision (AP), which means that it was able to correctly detect 42.4 out of 100 scene transitions. Once the videos were segmented at the scene granularity, the best hyperparameter configuration for the Video-text match model was: learning rate of 10^{-4}, 4 captions per video, and output feature space dimensionality for comparison of 512, 2 LSTM layers with input dimensionality of

300, hidden layer dimensionality of 512, and batch size of 4. The model overall performed well as this configuration retrieved the top 10% videos based on the similarity score with the text query with 85.32% accuracy. Therefore, once a text query is given to our model, it will likely retrieve the correct video scene in the top 10% of results with a total accuracy of **36.2%**.

This is our best-case scenario, as it performs worse on text queries it has not seen before, such as animals that are not present in the training dataset. A more robust dataset would help the model generalize better. Regardless, this is a significant advancement given that in a typical clip with 15 shots, the accuracy of randomly choosing the correct scene is less than 1% (the number of possible scenes is $\binom{15}{2} = 105$).

5.2. Future Work

Our current video-text matching model should still be considered preliminary work. The current implementation likely leaves significant performance on the table. As a result of hardware constraints and the intensive training process described in sections 3.3 and 5.2 there is still more potential in performing further hyperparameter tuning. In addition to the aforementioned learning rate tuning, there is a myriad of other factors that can be enhanced, both relating to the architecture itself and the training process. These hyparameters with room for further improvement include: tuning the feature space dimensionality, training captions per video, batch size, and modulating the number of specific layers in the model. These values were generally selected off loose heuristics developed from other related works, but additional fine-tuning is necessary to build a more robust architecture. Moreover, it would be valuable to be able to train the model on more powerful hardware, which would allow us to remove our current restrictions on model size, batch size, video frame rate, etc. Moreover, a more complex similarity calculation could be leveraged to further improve prediction results.

Qualitative analysis of the model also appears to show that the Vatex dataset is not nearly large or diverse enough to allow the model to generalize well. The dataset depicts typical human activities but under-represents other categories that may be of interest. For example, in one experiment, we ran the model against clips originating from a nature documentary and found that it struggled, likely due to a lack of clips from this domain in the dataset. Thus, it would be a useful extension to not only train the model on the entirety of Vatex but also to incorporate other similar datasets such as ActivityNet. We also think that introducing transformers to our model in place of LSTMs could offer up additional performance– specifically, in the case of the video data, where the original input sequences (before downsampling) are hundreds of frames long, transformers might do a better job at capturing long-term dependencies in our input sequences. One last extension could be the use of features extracted by our models in downstream tasks, like classification. Currently, it is difficult to identify where holes in our model lie, but if for example, our video features perform poorly on downstream tasks, but our text features perform well, then this could indicate that the gap lies in our video encoding.

Lastly, there would be value in including additional video information in the training data. Augmenting the video encoder to evaluate not only frame images but also closed captioning and audio data would result in further performance improvement. Overall, because this is the first work, to the best of our knowledge, attempting to accomplish scene retrieval from a text query there is plethoric scope for further enhancement.

References

[1] Jingyuan Chen, Xinpeng Chen, Lin Ma, Zequn Jie, and Tat-Seng Chua. Temporally grounding natural sentence in video. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 162–171, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. 1, 2

[2] Cisco. Cisco annual internet report(2018-2023), 2020. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf. 1

[3] Jianfeng Dong, Xirong Li, Chaoxi Xu, Shouling Ji, Yuan He, Gang Yang, and Xun Wang. Dual encoding for zero-example video retrieval. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9338–9347, 2019. 2

[4] Victor Escorciaa, Mattia Soldana, Josef Sivic, Bernard Ghanem, and Bryan Russellc. Finding moments in video collections using natural language. Feb. 2022. https://arxiv.org/abs/1907.12763v2. 1, 2

[5] J. Gao, Chen Sun, Zhenheng Yang, and Ramakant Nevatia. Tall: Temporal activity localization via language query. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5277–5285, 2017. 1, 2

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[7] Lisa Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. Localizing moments in video with natural language. pages 5804–5813, 10 2017. 1, 2

[8] Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. Localizing moments in video with temporal language. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1380–1390, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. 2

[9] Bingbin Liu, Serena Yeung, Edward Chou, De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. Temporal modular networks for retrieving complex compositional activities in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 2

[10] Sho Maeoki, Yusuke Mukuta, and Tatsuya Harada. Video moment retrieval with text query considering many-to-many correspondence using potentially relevant pair, June 2021. https://arxiv.org/abs/2106.13566. 1, 2

[11] Thang-Long Nguyen-Ho, Minh-Khoi Pham, Tien-Phat Nguyen, Hai-Dang Nguyen, Minh N. Do, Tam V. Nguyen, and Minh-Triet Tran. Text query based traffic video event retrieval with global-local fusion embedding. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3133–3140, 2022. 1, 2

[12] Anyi Rao, Linning Xu, Yu Xiong, Guodong Xu, Qingqiu Huang, Bolei Zhou, and Dahua Lin. A local-to-global approach to multi-modal movie scene segmentation. *CoRR*, abs/2004.02678, 2020. 3, 5

[13] Robert Sklar. *Film: An international history of the medium.* Thames and Hudson, 1990. 1

[14] Xiao Sun, Xiang Long, Dongliang He, Shilei Wen, and Zhouhui Lian. Vsrnet: End-to-end video segment retrieval with text query. *Pattern Recognition*, 119:108027, 2021. 2

[15] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 3

[16] Jing Xu, Yu Pan, Xinglin Pan, Steven Hoi, Zhang Yi, and Zenglin Xu. Regnet: self-regulated network for image classification. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 3

[17] Da Zhang, Xiyang Dai, Xin Eric Wang, Yuan fang Wang, and Larry S. Davis. Man: Moment alignment network for natural language moment retrieval via iterative graph adjustment. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1247–1257, 2019. 2

[18] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 3

A. Contribution Table

Group Member	Contributions
Corey Xing	Implementation and evaluation of video-text matching model Vatex data preprocessing Writing final report and poster
Daanish Mohammed	Training Scene Segmentation model creating inference script to enable easy integration of the models writing final report poster session
Junha Lee	Creating test videos for final model video-text matching model tuning Transformer encoding final report poster session
Shez Malik	Google Cloud setup video-text matching model tuning and training secondary shot segmentation model and inference script final report poster session