Article

Video Caption Based Searching Using End-to-End Dense Captioning and Sentence Embeddings

Akshay Aggarwal 1, Aniruddha Chauhan 1, Deepika Kumar 1,*, Mamta Mittal 2, Sudipta Roy 3,*, and Tai-hoon Kim 4,*

1 Department of Computer Science & Engineering, Bharati Vidyapeeth's College of Engineering, New Delhi 110063, India; akshayagrwl98@gmail.com (A.A.); aniruddhac7@gmail.com (A.C.); deepika.kumar@bharatividyapeeth.edu (D.K.)
2 Department of Computer Science & Engineering, G. B. Pant Govt. Engineering College, Okhla, New Delhi 110020, India; mittalmamta@gbpec.edu.in
3 PRTTL, Washington University in Saint Louis, Saint Louis, MO 63110, USA
4 School of Economics and Management, Beijing Jiaotong University, Beijing 100044, China
* Correspondence: sudiptaroy@wustl.edu (S.R.); taihoonkim@bjtu.edu.cn (T.-h.K.)

Received: 15 April 2020; Accepted: 8 June 2020; Published: 10 June 2020

Abstract: Traditionally, searching for videos on popular streaming sites like YouTube is performed by taking the keywords, titles, and descriptions that are already tagged along with the video into consideration. However, the video content is not utilized for searching of the user’s query because of the difficulty in encoding the events in a video and comparing them to the search query. One solution to tackle this problem is to encode the events in a video and then compare them to the query in the same space. A method of encoding meaning to a video could be video captioning. The captioned events in the video can be compared to the query of the user, and we can get the optimal search space for the videos. There have been many developments over the course of the past few years in modeling video-caption generators and sentence embeddings. In this paper, we exploit an end-to-end video captioning model and various sentence embedding techniques that collectively help in building the proposed video-searching method. The YouCook2 dataset was used for the experimentation. Seven sentence embedding techniques were used, out of which the Universal Sentence Encoder outperformed over all the other six, with a median percentile score of 99.51. Thus, this method of searching, when integrated with traditional methods, can help improve the quality of search results.

Keywords: video captioning; embeddings; deep learning; sentence embeddings

1. Introduction

Digital communication today is not only reliant on text but also on multimedia such as image, audio, and video. Video has become a popular way of communication between users, which has been helped by the increase in internet bandwidths and storage spaces. The increase in video data has led to an interest in the understanding of video for different applications such as video retrieval, surveillance, and online advertisements. Video retrieval is a significant task in the domain of video understanding for the simple reason that for the massive amount of video content present online, there have to be adequate mechanisms in place that can assist in the search and retrieval of relevant videos. Popular streaming websites like YouTube rely on metadata like keywords, titles, and the description of videos for their recommendation and search engines [1]. Retrieval of information is not done using video content. To insert a short video or GIF in a presentation, people usually make an online search to find the content. Occasionally the results are not congruent with the query. Since the tags are used for the searching of video clips and GIFs, there are not any mechanisms in place to retrieve relevant video
clips without appropriate tags or descriptions. This challenge of getting relevant video clips from a large video or a video not having good tags and description can be tackled by deep learning (DL). Recent advances in DL, primarily in the fields of image [2–5] and signal [6,7], have inspired researchers to come up with techniques to learn robust representations of features to leverage ample multimodal clues in video data.

We segment the task of video-searching into two parts. In the first part, the encoding of events in the video is done to get the captioned events of the video, and the second part deals with matching the captioned sentence with the query sentence of the user. So, we can look at the task under the lens of video captioning and similarity between generated captions and user queries. Video captioning is not a new research topic; before the advances in DL, hand-crafted features that detected the visual aspect of the video, using templates that generated fixed syntactical structures as sentences, were used to tackle this problem [8,9]. In contrast, DL-based video captioning systems employ sequence-learning-based methods for video captioning. Sequence-to-sequence models used in video captioning systems follow the encoder–decoder architecture. The encoder, using neural networks, learns video representations, and the decoder translates the learned video representation into a caption. The multimodal features that are extracted using the encoder are aggregated to generate a concise representation. Recent progress made in the architecture of deep neural networks has surged the use of Convolutional Neural Networks (CNNs) [10] for the encoding of visual features, and Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) for learning the sequential information in the videos [11–13]. After the captions are generated, the next task is to retrieve the video clips from the database that are relevant to the user. For this, the similarity between the generated caption and the user’s query is calculated using a distance or similarity metric. The videos in the database are then ranked according to the metric score, and the relevant results are returned. The output will be a sorted list with the videos that most closely resemble the query of the user at the top.

The main contributions of this research work are summarized as follows:

1. A new DL-based automated approach for video searching is proposed, where two different modules work in coalesce to obtain the relevant sorted results.
2. An end-to-end masked transformer-based dense video captioning model is used on a cooking-oriented dataset to perform the testing and experiments.
3. Seven different sentence-embedding techniques are tested for encoding the captions and the queries into the same embedding space. The cosine similarity metric is used to match the queries with captions, and the same metric is used to rank the results.

This paper is organized as follows: Section 2 discusses the literature review for video-captioning and different embeddings. After this, in Section 3, we briefly discuss various datasets available for the video-captioning task and then demonstrate the thorough analysis of the YouCook2 dataset. Then, we discuss the proposed methodologies for our two sub-tasks. The first is to obtain the captions for the video clips and then to demonstrate the performance of different embeddings for the search based on generated captions. Section 4 is the results section, where the results for video captioning and searching at different percentile levels are discussed. Then, Section 5 is the concluding section of the work, where the future prospect of the method is discussed.

2. Related Works

In this section, we first discuss different strategies for building a video-caption generation model. After this, we provide an overview of the word and sentence embeddings and how they can be compared in a common space.

**Video Captioning:** Video captioning is amongst the newest problems that have gained the attention of researchers from both computer vision (CV) as well as natural language processing (NLP) communities. The objective of video captioning is to automatically generate a sequence of words that is complete and is in natural language, based on the content of the video [14]. This problem can
be thus formulated: given an input video $V = [v_1, \ldots, v_N]$ where $v_i$ denotes the $i^{th}$ frame of the video sequence, generate a word sequence $W = [w_1, \ldots, w_T]$ where $w_t$ denotes the $t^{th}$ word of the generated caption. A good caption generation model should be able to capture how objects, activities, and scenes present in a video relate to each other and formulate this relation into a meaningful sentence, so the task is undoubtedly very challenging. The task is commonly divided into two subtasks: encoding the video sequences, and the caption generation where encoded video sequences are processed to form an eloquent sequence of words. Figure 1 illustrates this DL-based encoding–decoding architecture for video captioning. Typically, the encoder part is based on CNN architectures and the decoder part is based on RNNs.

![Figure 1. Deep-learning (DL)-based video captioning architecture.](image)

**The Encoder—Features for Video Captioning:** With the promising development in DL in recent years, it has been able to show exceptional performance in resolving several artificial intelligence problems. DL-based spatial algorithms, such as 2D and 3D CNNs, are exploited to improve state-of-the-art video representation [15–17]. The task of extracting video representation is generally divided into two steps: multimodal feature extraction and feature aggregation. Mainly, there are four types of modalities that are important to build a video understanding model: visual, audio, motion, and semantic. Many researchers have been able to apply DL-based state-of-the-art methods to extract features from these modalities. Briefly, for visual feature extraction, CNN architectures such as VGG Net and Resnet are the most popular choices. For obtaining fixed-length audio features, Mel Frequency Cepstral Coefficients (MFCC) and Bag-of-Audio-Words are the broadly adopted approaches [18]. 3D CNNs have been applied to capture motion within videos by assuming a video sequence as a series of frames stacked together to form a 3D image. At last, to capture other semantic features explicitly, researchers have incorporated LSTMs as well [19]. As the features are often obtained from different modalities and can have variable shape or length, therefore, it becomes crucial to have a suitable method that can aggregate them into a fixed-length representation. One way is to generate an encoded state of the feature sequences by passing them through LSTM/GRU. However, in this way of aggregation, the contribution of all the features is the same to the decoder, which is not feasible in a practical scenario. Hence, researchers have developed algorithms that can dynamically learn and assign weights to these features; these algorithms are often referred to as temporal attention [20,21]. Intuitively, it can be concluded that spatial/visual feature is still the most important feature for video captioning, so it also becomes essential to have a method that can dynamically indicate objects of interest in the different spatial regions in the video. To do this, the Multi-level Attention Model-Recurrent Neural Network (MAM-RNN) devised an approach that incorporates weights from the previous frame to compute spatial weights for a particular frame [22]. To distinguish foreground from background, the Spot and Aggregate Module (SAM) calculates saliency scores that result in binary maps plotted according to particular threshold values [23]. As it has become ubiquitous to have multiple features for video captioning, sometimes strategies as simple as concatenating all features have worked quite well [24]. To incorporate dynamic weight assignment in this feature-concatenation strategy, attention mechanisms are applied to different modalities so that the contribution of each feature is different to the decoder [25].
Intuitively, one can also say that captions that are similar in context should represent similar kinds of video content. Based on this assumption, Gkountakos et al. proposed yet another encoder–decoder-based architecture where, first, the words in the vocabulary are converted to word embeddings, and then, the vocabulary is mapped to specific clusters using the K-Means clustering algorithm [26,27]. Additionally, the authors also propose a penalty/reward-based loss function in order to make the architecture agnostic of the CNN-based feature extractor and the dataset. This further assures that their proposed modeling can be applied with any baseline architecture.

**The Decoder Caption Generation:** In caption generation, the main objective can be formulated mathematically, as given the original caption $Y = \{y_1, \ldots, y_T\}$, and the generated word probabilities $W = \{w_1, \ldots, w_t\}$ minimize the log-likelihood function $L$.

$$L = \sum_{t=1}^{T} L_t(w_t, y_t)$$

(1)

$$L_t = -\sum_i y_i \log w_i$$

(2)

As much as it is essential for the generated captions to suitably represent the video content, more recent developments are now focusing on refining the quality, i.e., making the generated captions more fine-grained and diverse [28]. For this, Xiao et al. incorporated convolutional architecture to the LSTM-based decoder to generate fragment-level features. These features help in capturing the information cues from local motion in the video. For improvement in the quality of captions, Pan et al. and Gao et al. adopted a straightforward way of projecting the video features and sentence embeddings in the same space and used an optimization algorithm to minimize the differences between the two [29,30].

Xiao et al. highlighted the limitations of using traditional LSTM-based models [31]. These models, despite being adaptive, are unable to maintain a good level of performance while transmitting the semantic information. This leads to the generation of poor-quality captions, which can either be incomplete or made up of repetitive words. To overcome this limitation, the authors proposed a text-based dynamic attention model (TDAM) which utilizes hierarchical LSTM for next-word generation. You et al. developed semantic attention (SA) architecture for image captioning [32]. Their attention-based architecture combines top-down (image-to-words) and bottom-up (words-to-image) strategies and obtains rich semantic information from the images for generating captions. One thing to note is that the authors have used the nearest-neighbor algorithm for retrieving visual similarities in the images of the dataset, so the ability of the model may be restricted to the particular dataset.

**Word and Sentence Embeddings:** Word and sentence embeddings are popular, and to some extent, are a universal way for representing words and sentences by fixed-length encoded vectors that can assimilate the general semantic relationships of the text. The main advantage of this way of representation is the drastic improvement in the processing of textual data. With the availability of huge textual data around the web, researchers are more inclined towards finding embeddings that can be applied universally, i.e., embeddings that are pretrained on some huge corpus and then use the same embeddings on some other downstream task, such as classification or to build a question–answer system. For embeddings, it has been normal for the past few years to have a distributional hypothesis-based unsupervised way of word representation, word2vec and GloVe being the most common approaches [33,34]. Word2vec is a feedforward skip-gram-based model that is trained for predicting the context words, given an input word. Once trained, the sentence can be put into the model word-by-word, and corresponding word-embeddings can be obtained. GloVe, on the other hand, is based on a matrix factorization technique on the word-context matrix to find a low-dimensional representation. FastText, a universal word embedding, is an extension of word2vec, which is also responsible for boosting the recent interest in language-model development [35]. The main advantage of FastText over word2vec is that it can generate representations for the words that were not
there in the vocabulary while training. The authors of FastText have made their vectors available in 157 languages. These vectors have been trained on Crawl and Wikipedia. One of the most anticipated developments for the word-embedding is ELMo [36]. Similar to FastText, ELMo is also capable of computing representations for words that are out of vocabulary. This is due to the fact that input to ELMo are characters instead of words. ELMo uses a bidirectional language model to represent a word as a function of sentences from the entire corpus. An approach similar to that of ELMo was employed to create ULMFiT [37]. After ULMFiT, another approach that could learn contextual word embeddings by leveraging more training data and using novel training methods known as Bidirectional Encoder Representations from Transformers (BERT) was released [38].

**Sentence Embeddings:** Many competing approaches for sentence embeddings have emerged in the past few years. Some procedures include training in a supervised manner, while others are trained in an unsupervised manner, as well as multitask learning. Mainly, there are four types of strategies that are being studied the most: simple-averaging-based, supervised, unsupervised, and multitask learning scheme. In a simple-averaging-based method, all the word vectors for the words from a sentence are encoded using the bag-of-words approach and then averaged to compute the sentence’s embedding. Although this approach seems simple, it has facilitated in developing some robust baseline approaches. One of these was developed by Arora et al. [39], where they proposed to use any popular word-embedding and then encode the sentence as a weighted-combination of the word vectors produced using embeddings. After that, a common component removal is performed to generate a final sentence embedding; this approach was termed smooth inverse frequency (SIF).

Similar to the skip-gram model for word embeddings, the skip-thoughts vector is an unsupervised approach that is trained for predicting surrounding sentences based on an input sentence [40]. Logeswaran et al. reformulated this task of sentence prediction into a classifier task where the next sentence is chosen from a set of candidates and called their method quick-thoughts vectors [41]. Ethayarajh et al. built upon the work of Arora et al. to create an unsupervised method, which authors call unsupervised SIF (uSIF) for creating sentence embeddings that did not require any hyperparameter tuning [42]. InferSent, a recent supervised approach, trained a classifier on Stanford Natural Language Inference (SNLI) corpus that contains around 570,000 sentences of three categories [43]. Sentence-BERT decreased the time for finding similar sentences using BERT/RoBERTa from 65 h to a meager 5 s in comparison; this was done by modifying the pretrained BERT network [44]. In multitask learning, the core purpose is to combine multiple training objectives in a generalized way to generate an output. Universal Sentence Encoder (USE), recently released by Google, built a multitasking encoder that is trained on various data sources to perform multiple tasks to accommodate a generalized mechanism for a wide variety of natural language understanding tasks [45].

### 3. Proposed Methodology

To model the caption-based search through video clips, the first underlying step is to obtain the relevant captions for the video clips. In this section, the datasets available for video captioning are discussed, after which the YouCook2 dataset is analyzed. Subsequently, we discuss the requirements that are taken into consideration during the selection of the video-captioning model. Then, based on these requirements, a dense video-captioning model is selected and its ability exploit to incorporate it into the proposed approach for video searching. All the analysis and experiments presented in this and the upcoming sections are performed using the computational resources provided by Google Colab; for training purposes, the authors used the 12 GB Tesla K80 GPU environment provided by Google Colab.

#### 3.1. Dataset Description

There are several datasets that have been made available for this research problem. Some datasets are confined to some specific domains; for instance, YouCook2 is a cooking-task-oriented dataset containing 2000 untrimmed videos in third-person view downloaded from YouTube without any
constraints on camera [46]. Each of the videos has corresponding English sentences describing the cooking procedure for specific recipes. The MPII Movie Description (MPII-MD) dataset consists of movie snippets aligned with the audio description [47]. This dataset contains around 68,000 sentences and snippets of video from 94 movies. MSR-VTT, short for MSR Video to Text, released by Microsoft Research, provides 10,000 clips and 200,000 clip-pair sentences from different categories, including movies, TV shows, people, music, and sports [48]. MSR-VTT currently has the most extensive vocabulary compared to other datasets. ActivityNet 200 (Release 1.3) contains 10,024 training, 4926 validation, and 5044 testing videos, totaling around 20,000 videos from 200 activity classes like eating and drinking, recreation, and household activities [49]. Table 1 summarizes the statistics for these datasets. Average words denote the average number of words per sentence, and the number of words is the vocabulary size.

YouCook2 is the newest dataset amongst the four datasets shown in Table 1. Unlike other datasets where annotations are limited to the actions performed, YouCook2 provides annotations with procedure segments that contain much more semantic information. Procedure segments are able to capture human-involved processes, as well as the background activities, in a better way. In addition, since the vocabulary size (number of words) of the YouCook2 dataset is comparatively less when compared to the other video-captioning datasets and the videos are also confined to a specific domain, it makes sense to initially select such a dataset for testing the proposed approach. Hence, all further experiments are performed on the YouCook2 dataset.

It can be seen from Figure 2 that most of the annotations contain names of common cooking ingredients, such as salt, water, pepper, oil, sauce, and common cooking utensils, such as pans and bowls. It is also interesting to notice that the common activity performed during cooking is limited to actions such as add, place, mix, stir, and more. The different sizes of words in the figure signify how frequently certain words are used in the annotations provided in the dataset; the larger the size of a word, the more frequent is its use.

Further, it can be seen from Figure 3a that length for most of the videos in the dataset is only 3–6 min long. Almost all the videos are divided into multiple segments, with most videos having around 8 segments. It is also worth noting that the number of words per annotation of a video stays typically in the range of 50–150 words and hardly crosses 200. As each of the videos has been divided into segments, most of the segments have been annotated with very few words, mostly below 10 words.
Figure 3. Statistics of annotations provided by the YouCook2 dataset. (a) Duration of videos in seconds. (b) The number of segments per video. (c) Number of words in the annotations of a video. (d) Number of words in the annotations of a video-segment.

3.2. Data Preprocessing

The data preprocessing step involves caption generation. The captions are obtained for all the 457 videos present in the validation set split provided by YouCook2, using the available model. Initial data preprocessing steps include downsampling all the videos by 0.5 s. After downsampling, to feed these videos to the video encoder, two types of features are extracted—appearance features that are obtained by passing the frames through ResNet-200 and optical flow features that are obtained using BN-Inception [50]. Both of these networks were pretrained on the ActivityNet dataset. The authors had additionally set the limit for window size per video to be 480 frames; therefore, the length of all the videos is either padded with zeros (in the case where the number of frames is less than 480) or clipped in case the frames are more than 480. To obtain the captions, the feature vectors for each of the videos can be fed into the trained model, with all the parameters set to nontrainable.

3.3. Generation of Video Captions

As discussed in the literature review, available video-captioning models are composed of two submodules, encoder and decoder. Almost all of the past approaches train these two modules separately and then combine their learned features to work at the task of video captioning. However, training these modules separately does not take into account the influence of these two modules on each other and thus declines the accuracy of generated captions. Therefore, our first requirement is to select the model which is end-to-end, i.e., the two modules are trained in a coordinated and continuous way, and not separately, so as to obtain more plausible captions.
Secondly, in dense captioning, particularly for the decoder submodule, the generated descriptions are relatively longer in terms of the number of tokens (words); thus, it becomes crucial to consider the importance of learning the representations that can sustain the information for a more extended time period. Recurrent neural networks (RNNs) are a suitable and popular approach for sequence modeling but often fail to incorporate the long-term dependencies. LSTMs and GRUs are other variants of RNNs that are explicitly designed to solve this problem. However, more recently, much faster and better models—transformer-based attention models—have potentially improved the benchmark scores on several NLP tasks. These attention-based models are also capable of learning long-term dependencies. Henceforth, taking these two requirements as our selection criteria, we select the masked transformer-based end-to-end dense video captioning model, which was developed by Zhou et al. [51]. Now, we briefly discuss the components of this model and how the trained model was utilized in obtaining the captions for the YouCook2 dataset.

The three components of this model are video encoder, proposal decoder, and captioning decoder. The function of the video encoder is the same as discussed above, i.e., to encode the series of video frames into a feature vector space. In this case, the encoder is a CNN-based layered network with a ReLU activation function. Along with CNN, self-attention is also applied to improve the context-learning ability of the encoder. The second component, the proposal decoder, takes the encoded feature representation from the encoder as an input and uses different anchors to output the event proposals. Event proposals are nothing but the starting and ending time for a particular event and the corresponding confidence score. The proposal decoder for this model is based on ProcNets, which uses an explicit anchor-based mechanism and 1D CNNs on encoded features for obtaining the event proposals. An event proposal is represented using a tuple: \((s, e, p)\), where \(s\) and \(e\) are the starting and ending time or boundaries for the event, and \(p\) is the associated probability score where \(p \in [0, 1]\).

The third and last component of the model is the captioning decoder. The captioning decoder is responsible for generating the word tokens by taking visual features from the video encoder and event proposals from the proposal decoder as the inputs. To ensure the end-to-end training of all three components in a combined way, an additional differentiable masking scheme is applied in the captioning decoder. Figure 4 depicts the proposed searching approach. “Embed” blocks in the diagram correspond to the different sentence embeddings that are experimented with to convert the generated captions and the queries in the same embedding space.

**Figure 4.** Methodology diagram.

### 3.4. Caption Search and Sentence Embedding

The search of video captions is performed by embedding both the captions generated and the query captions in a similar embedding space, where the queries are compared with the captions.
The comparison is made based on the similarity metric; cosine similarity in this case. The caption embedding that is more similar to the query embedding is more likely to be the result being searched for. There are no datasets available for searching for video clips from captions in a manner that is proposed in this paper. In order to test our methodology, the test set of captions of videos were used as the queries for the search of videos. The test captions were treated as the queries, and the predicted captions were treated as the search space for the query. The search is considered successful if the video to which a test caption belongs is present in the result set of the set of videos containing the predicted captions similar to the test caption. To rank the videos that are most similar, cosine distance is used. The evaluation of the search is done using a percentile metric.

4. Results and Analysis

To assess the task of video captioning quantitatively, Bleu@N, METEOR, and CIDEr scores are the most commonly adopted evaluation metrics [52–54]. BLEU, short for Bilingual Evaluation Understudy, is a score that was originally developed to evaluate translations but is now ubiquitously used to assess most text-related tasks. It measures the fraction of candidate’s N-grams that match with the references’ N-grams and returns a corresponding score between 0 and 1. For instance, if the value for N is 4, then for computing Bleu@4, candidate sentence and the reference sentence are compared with each other by taking 4 words at a time. In our case, we compute the BLEU score for four values of N = 1, 2, 3, and 4. As for Bleu@1 (only one word at a time is compared for calculating the score), it is inevitably higher when compared with the BLEU score for the higher values of N. Despite being easy and fast to calculate, it has some major drawbacks: the meaning of sentences is not taken into account, morphologically rich languages are not handled well, and the score does not map well to human judgments. So, METEOR, short for Metric for Evaluation of Translation with Explicit Ordering, is often used along with BLEU as it computes the harmonic mean of unigram precision and recall and extends the capability by including similar words and stemmed tokens while matching. CIDEr, short for Consensus-based Image Description Evaluation, is the newest amongst the three metrics and performs term frequency inverse document frequency (TF-IDF) weighting for the N-grams.

After feeding the video vectors obtained during the data preprocessing step to the captioning model, the model outputs the caption and the metric scores at different threshold values for tIoU. tIoU here denotes the amount of overlap between the proposed segment and the ground-truth segment, and the metric scores are only computed for generated and corresponding ground-truth sentences if the tIoU value is higher than the threshold; otherwise, it is 0. Table 2 below summarizes the values for metric scores on the validation set at different thresholds.

<table>
<thead>
<tr>
<th>Metric</th>
<th>tIoU 0.3</th>
<th>tIoU 0.5</th>
<th>tIoU 0.7</th>
<th>tIoU 0.9</th>
<th>Average Score across All tIoUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIDEr</td>
<td>16.64</td>
<td>18.57</td>
<td>21.65</td>
<td>33.84</td>
<td>22.67</td>
</tr>
<tr>
<td>Bleu@4</td>
<td>1.43</td>
<td>1.35</td>
<td>1.16</td>
<td>0.70</td>
<td>1.16</td>
</tr>
<tr>
<td>Bleu@3</td>
<td>4.36</td>
<td>4.20</td>
<td>3.66</td>
<td>2.74</td>
<td>3.74</td>
</tr>
<tr>
<td>Bleu@2</td>
<td>10.72</td>
<td>10.84</td>
<td>10.45</td>
<td>8.06</td>
<td>10.02</td>
</tr>
<tr>
<td>Bleu@1</td>
<td>24.15</td>
<td>24.29</td>
<td>23.87</td>
<td>19.40</td>
<td>22.93</td>
</tr>
<tr>
<td>METEOR</td>
<td>9.16</td>
<td>9.28</td>
<td>9.16</td>
<td>7.31</td>
<td>8.73</td>
</tr>
</tbody>
</table>

Using end-to-end masked transformer dense video captioning on the YouCook2 dataset for generating captions on the validation set yielded us with 36,977 unique captions. The validation set contains 457 videos that have 3492 segments and captions. The captions found in the validation set are what are being used as search queries. Accordingly, \( n_{\text{video}} \) is the number of unique video captions generated by the model, which is equal to 36,977.
Cosine similarity is a popular metric used to find the similarity between two vectors. If \( A \) and \( B \) are two vectors, their cosine similarity score, \( \text{similarity}_{\text{cosine}} \), can be calculated, as shown in Equation (3). Cosine distance, \( \text{distance}_{\text{cosine}} \), can then be derived from the cosine similarity score, as shown in Equation (4). In our case, we compute \( \text{distance}_{\text{cosine}} \) between the generated captions and the captions provided in the validation set in order to sort the results.

\[
\text{similarity}_{\text{cosine}} = \cos \theta = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}
\]

\[
\text{distance}_{\text{cosine}} = 1 - \text{similarity}_{\text{cosine}}
\]

Due to the absence of a dataset that provides the relevant video clips for a query search, the test captions and percentile metric are used to evaluate the search. The percentile metric for measuring the performance of our task \( P(n_{\text{video}}, t_{\text{video}}) \), is defined as

\[
P(n_{\text{video}}, t_{\text{video}}) = 100 \times \frac{|n_{\text{video}}| - \text{rank}(t_{\text{video}})}{|n_{\text{video}}|}
\]

where \( n_{\text{video}} \) is the entire dataset of clips present in the prediction set, \( t_{\text{video}} \) is the predicted video clip that matches the video to which query belongs to, and \( \text{rank} \) outputs the position of a caption in \( n_{\text{video}} \) sorted by probability of match of video. On the basis of this evaluation metric, Top-1% performance refers to the percentage of captions in our dataset with percentile at least 99; similarly, Top-10% performance refers to the percentage of captions in our dataset with a percentile of at least 90. Median refers to the median percentile of the queries, i.e., the minimum percentile in which the search is successful in half the queries. The results obtained are presented in Table 3. It depicts the different sentence embedding models used to encode the captions and queries in the same space at different percentile levels.

**Table 3.** Comparative analysis of different sentence embedding models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Top-1%</th>
<th>Top-5%</th>
<th>Top-10%</th>
<th>Top-15%</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIF-Glove</td>
<td>58.53</td>
<td>88.32</td>
<td>95.79</td>
<td>98.45</td>
<td>99.34</td>
</tr>
<tr>
<td>SIF-FastText</td>
<td>58.42</td>
<td>88.66</td>
<td>95.70</td>
<td>98.25</td>
<td>99.31</td>
</tr>
<tr>
<td>uSIF-Glove</td>
<td>61.74</td>
<td>90.38</td>
<td>97.28</td>
<td>98.85</td>
<td>99.44</td>
</tr>
<tr>
<td>uSIF-FastText</td>
<td>63.55</td>
<td>90.78</td>
<td>97.36</td>
<td>98.94</td>
<td>99.45</td>
</tr>
<tr>
<td>Sentence-BERT</td>
<td>60.85</td>
<td>90.52</td>
<td>96.99</td>
<td>98.80</td>
<td>99.37</td>
</tr>
<tr>
<td>Sentence-RoBERTa</td>
<td>62.23</td>
<td>91.27</td>
<td>97.77</td>
<td>98.45</td>
<td>99.42</td>
</tr>
<tr>
<td>USE</td>
<td><strong>65.01</strong></td>
<td><strong>92.30</strong></td>
<td><strong>97.77</strong></td>
<td><strong>99.11</strong></td>
<td><strong>99.51</strong></td>
</tr>
</tbody>
</table>

The results show that USE performs better than all the other models, which include SIF and uSIF that use both GloVe and FastText embeddings, Sentence-BERT, and Sentence-RoBERTa. USE outperforms all the other models in all respects of the experiment. In addition, it can be seen that the greatest deviation in scores is in Top-1% and it later tapers off as we move to Top-5% and Top-10%, it signifies that the top search results are better found with USE and as we increase the number of search results, the other methods come closer. Besides, the 65.01 score for Top-1% denotes that for all the search queries, 65.01% of the times, the relevant video clip is within the Top-1% of the search results. From this, it is inferred that of the 457 videos from which we are searching for a query, at least 65% of the time, the video we are searching for will be present in the Top-1%. The total number of queries that were searched is 3492; from these queries, 65% of the time, the video being searched is present in Top-1%, which translates to the relevant video being present in the top 5 search results out of total 457. The score of the different percentiles (such as Top-1%) signifies the accuracy of our model in listing the correct video in that particular percentile.

The percentile of every query or test caption in the validation set is noted for different models. As can be seen from Table 3 itself, for each of the embedding models, in approximately 99% of the
cases, the relevant search results are within the Top-15%. Thus, to visualize these results, we further clip the percentile level to be greater than 85, as shown in the plots at the right-hand side in the figure below. The histograms are made using the stored percentiles, which are represented in Figure 5.

![Percentile Distribution SIF - GloVE](image1)

![Percentile Distribution SIF - GloVE (Percentile>85)](image2)

![Percentile Distribution SIF - FastText](image3)

![Percentile Distribution SIF - FastText (Percentile>85)](image4)

![Percentile Distribution uSIF - GloVE](image5)

![Percentile Distribution uSIF - GloVE (Percentile>85)](image6)

![Percentile Distribution uSIF - FastText](image7)

![Percentile Distribution uSIF - FastText (Percentile>85)](image8)

**Figure 5. Cont.**
5. Conclusions and Future Perspective

In this paper, the authors have proposed a new approach of searching through videos, where instead of the metadata associated with the video, the video content is exploited to obtain and rank the search results. To perform the search and demonstrate the results, this study was divided into two subtasks, where the first subtask objective is to generate video captions for the videos provided in the YouCook2 dataset. For this, an end-to-end video captioning model was used to encode the content of the video. In the second subtask, seven different embedding models were used to embed the captions and queries in a multidimensional vector space. Furthermore, a cosine similarity metric was used for matching the queries and captions. The results of all seven embeddings at different percentile levels were demonstrated, and it has been concluded from the results that the Universal Sentence Encoder outperforms all the other embedding models in all aspects, and, most significantly, outperforms them in the Top-1%, i.e., the accuracy for the top five search results. As can be seen from the results based on the evaluation metric scores, this way of searching gives promising performance and, if incorporated into existing video searching methods, can help improve the quality of search results. Furthermore, it can be helpful in searching unlabeled videos and searching for particular segments or events in...
longer videos. In the future, the same approach can be tested upon other video captioning datasets. In addition, a dataset of annotated videos and search results for videos, along with search queries, will aid in further research in this direction. Different video captioning models, embedding models, and similarity metrics can also be analyzed and experimented with to improve performance.

**Author Contributions:** Conceptualization, A.A. and M.M.; Formal analysis, D.K.; Funding acquisition, T.-h.K.; Investigation, A.A., D.K. and S.R.; Methodology, A.A. and A.C.; Project administration, M.M. and T.-h.K.; Resources, D.K., M.M. and T.-h.K.; Software, D.K. and M.M.; Supervision, D.K. and S.R.; Validation, A.C.; Visualization, A.C.; Writing—original draft, A.A. and A.C.; Writing—review & editing, S.R. and T.-h.K. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Conflicts of Interest:** The authors declare no conflict of interest.

**References**


© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).