applied sciences

MDPI

Article

A Smart System for Text-Lifelog Generation from Wearable Cameras in Smart Environment Using Concept-Augmented Image Captioning with Modified Beam Search Strategy †

Viet-Khoa Vo-Ho, Quoc-An Luong *, Duy-Tam Nguyen, Mai-Khiem Tran and Minh-Triet Tran *

Information Technology and Software Engineering Lab, VNUHCM—University of Science, Ho Chi Minh 800010, Vietnam; vhvkhoa@selab.hcmus.edu.vn (V.-K.V.-H.); ndtam@selab.hcmus.edu.vn (D.-T.N.); tmkhiem@selab.hcmus.edu.vn (M.-K.T.)

* Correspondence: lqan@selab.hcmus.edu.vn (Q.-A.L.); tmtriet@hcmus.edu.vn (M.-T.T.) Tel.: +84-983-118-326 (Q.-A.L.)

† This article is an extended research of our previous work "*Personal Diary Generation from Wearable Cameras with Concept Augmented Image Captioning and Wide Trail Strategy*", awarded as "Best Paper" in Symposium on Information and Communication Technology conference (SoICT 2018), DaNang City, Viet Nam, 6–7 December 2018.

Received: 2 March 2019; Accepted: 29 April 2019; Published: 8 May 2019

Featured Application: Our work can be applied as an IoT system to capture important events in daily life for later storage. From wearable devices with camera such as smart glasses, photos of events can be periodically taken and processed into description in text format. The description is then stored in a database on server and can be retrieved via another smart device such as smartphone. This let users easily retrieve the information they want for sharing or reminiscence. The descriptions of photos taken each day can also be gathered as a diary. Furthermore, the database is also a huge resource for analyzing user behavior.

Abstract: During a lifetime, a person can have many wonderful and memorable moments that he/she wants to keep. With the development of technology, people now can store a massive amount of lifelog information via images, videos or texts. Inspired by this, we develop a system to automatically generate caption from lifelog pictures taken from wearable cameras. Following up on our previous method introduced at the SoICT 2018 conference, we propose two improvements in our captioning method. We trained and tested the model on the baseline MSCOCO datasets and evaluated on different metrics. The results show better performance compared to our previous model and to some other image captioning methods. Our system also shows effectiveness in retrieving relevant data from captions and achieve high rank in ImageCLEF 2018 retrieval challenge.

Keywords: lifelog processing; image captioning; IoT system

1. Introduction

People usually want to keep footage of the events that happen around them for many purposes such as reminiscence [1], retrieval [2] or verification [3]. However, it is not always convenient for them to record those events because they do not have the time or tool at that moment. People also could miss some events because they do not consider those events important or worth keeping until later. With the development of technology, especially IoT system, smart environment such as smart home and smart office can be established and give people easy access to ubiquitous service. In a

smart environment, people can easily upload and download data to the cloud server using just a small device. Motivated by this, we found it is possible and necessary to develop a system that can help people automatically capture the events happening around them as personal lifelog.

The method to capture and store lifelog data is the key issue in this problem. Lifelog data [3] can be stored in various formats such as text, audio, photo, video or biometric data. Each format can be collected in various ways. Biometric data format can be collected from sensors of smart device such as smart phones, smart watches or activities trackers. A generic framework [4] is also developed for continuously recording information from mobile phones. Regular recorder or recorder in smart phones can record daily conversation as audio format data. Visual format such as photo or video can be taken from wearable or regular camera.

Visual format has the most potential because it is rich in information and easy to collect. However, raw visual data consume large storage capacity and it is difficult for users to conduct a query of a specific event. When retrieving a certain event, large amount of images needs to be analyzed to find the matching image. One common approach is breaking the image into set of entities appear in the image using different object detectors [5,6]. Another approach is to describe the content of the image into text format [7]. Retrieving based on text format is a lot easier than visual format. Therefore, we propose a framework that can generate description in text format from images taken from wearable cameras. Given an image taken from users' device, the image is then processed into a sentence that describes the content of the image. In other words, the data in visual format are transferred into text format but still maintain the information. Users can easily retrieve the information they need via simply text retrieval method. Furthermore, text format will also reduce the consumption of storage capacity. This helps save resources as well as increases the amount of events that can be kept.

However, generating description from image is a challenging problem due to the complexity of the image content. There has been many approaches proposed for this problem. In 2010, Farhadi et al. [8] proposed using a triplet of (*object*, *action*, *scene*) to represent the image and retrieve a caption from a set of sentences. In work of Kulkarni et al. in 2011 [9], a *Conditional Random Field* (CRF) is used to represent the relationship between objects and attributes. Words describing objects and attributes are then filled into a template sentence based on their relationship. Yatskar et al. [10] added generative grammar in their work to generate better captions. Inspired by methods in machine translation, Vinyals et al. [11] treated the image as "a special language" and used an encoder-decoder model to "translate" it into caption. The encoder uses a Convolutional Neural Network to extract features from images and the decoder uses a Recurrent Neural Network to generate captions. Karpathy et al. [12] further improved this method using a LSTM model [13].

Some recent works show breakthrough in the image captioning problem. In 2015, Kelvin Xu et al. [14] proposed a model using the attention mechanism. This mechanism is based on human's ability to focus on certain part of the image and extract local features. During generating the caption, the model can focus on the region that is relevant to the current generated word. In 2016, Quanzeng You et al. [15] applied the attention mechanism on conceptual tags extracted from detectors. In 2018, Peter Anderson et al. [16] used attention on regional feature to help the model focus on meaningful part of the image. We follow these work to develop an image captioning model for our system. Our contributions in this article are as follows:

- We propose a system for automatically capture footage of users' daily event and convert the collected images into text format via image captioning method. The system will enable users to keep track of special events in their daily life. The system will keep the information of the events in text format, helping save storage capacity. We also develop smart glasses with cameras. This device not only can easily capture images automatically for processing but also is wearable and fashionable.
- We also propose two improvements in the image captioning method. The first is using more tags to enhance the information input into the caption generator module. The second is adding a new criterion for selecting longer caption during beam search strategy.

In more details, based on our previous method presented in [17], we propose an improvement in our captioning method for better generating the lifelog description. In our previous method, images are processed through a feature extraction module and a tags extraction module. In the feature extraction module, we use a convolutional neural network to extract features from the image. In the tags extraction module, an object detector to extract names of the objects that exist in the images. The two kinds of features are then processed with attention mechanism similar to the models in [14,15]. The combination of the two features will be fed into a LSTM model [13] to generate captions. In this work, we consider replacing the model in the tags extracting module with a different model. We use an object detector with more tag names than the previous one.We also add a new criterion into the beam search strategy of the previous work. In our previous work, the caption will be selected via a beam search strategy that ensures the model to select complete caption. In this work, we add a criterion that prefers longer captions because long captions have more potential to describe the contents in more details.

We trained and tested our image captioning method on MSCOCO dataset. The results were evaluated on different metrics and compared to some previous methods. Our method shows comparable performance to other methods. We also evaluated the application of our system. The results show high potential for real life application.

Details of our system and the image captioning method are described in Section 2. In Section 3, we present the results on the dataset compared to some other methods. Discussion and conclusions are presented in Sections 4 and 5, respectively.

2. Materials and Methods

2.1. System Overview

Our system contains three main part: a wearable camera to capture images of daily activities, an image captioning module to transfer the image into description in text format and a database to store the description for later retrieval. Figure 1 illustrates the overview of our proposed personal lifelog generating system. We transmit the images captured from user's wearable camera to a sever periodically. The images can also be stored in a storage device such as an SD card, and then loaded to the server. The next step is generating image captions. We use our model to process each image and produce corresponding caption. With GPU support, about 15–20 images can be processed per second. The captions are stored instead of the images in a database for lower storage capacity. Users can easily retrieve the information of activities and objects by comparing the captions and the queries. In the case of events that users want to keep image for reminiscence, users can define some certain descriptions of what they want to keep. The events that have captions that match the descriptions will be highlighted and images of that special events can be kept for advanced retrieval. Please refer to Figure 1.

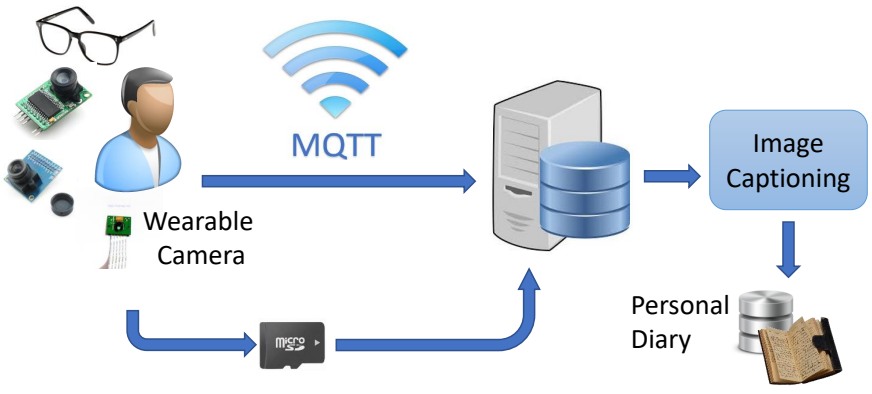

Figure 1. An overview of our lifelog generating system.

2.2. System Implementation

2.2.1. Smart Glasses With Camera

Our intention is to develop a device for users to wear in order to capture their lifelog. This requires three main properties: good design, low energy consumption and stable transmission. The device needs to be small and easy to carry. Users may feel uncomfortable carrying a heavy device or if the device makes them look not pretty. Because the user needs to use the device for a long time, it needs to have enough power to last until the users are available for charging. Because of small design and low energy consumption, directly processing data in the device would be impossible. This leads to the need of a stable transmission method to transmit data to the server for processing and storage. In the implementation of our system, we use smart glasses with camera to capture the image. The smart glasses have two main advantage: power consumption and data transmission. They are also easy to wear and can be used as a fashion accessory. We handle the processing tasks in smart glasses with Raspberry Pi Zero W. This module has a Broadcom BCM2835 CPU (up to 1 GHz) and 512 MB RAM. We also have stretch-lite OS installed on board. The OS we use is Raspbian, which supports all versions of Raspberry Pi. It is also the official operating system recommended by Raspberry producers. Besides Raspbian, other operating systems such as Debian, Yocto or Windows IoT can also be used. We choose to turn off the Tvservice, which is also called the HDMI status. This status is the status of the signal from Raspberry Pi to the HDMI gate. When turned off, it will help save energy for the system. We use 5 V power supply through GPIO instead of USB, which can bypass some energy-consuming part such as USB hub or registers and save energy for the system. With these configurations, the amount of energy consumption can be dropped from 80 mAh to 66 mAh. Please refer to Figure 2.

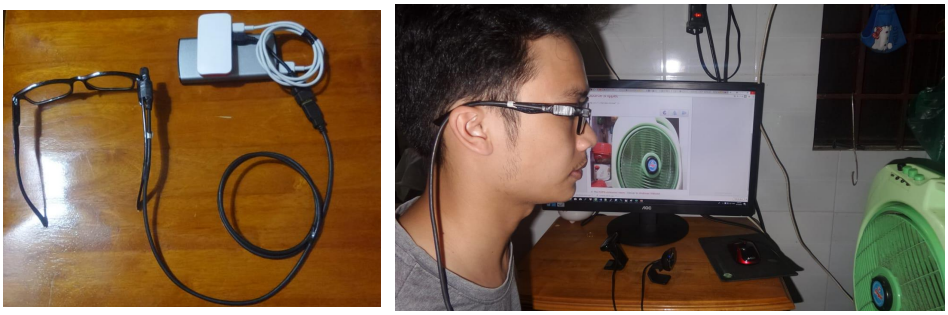

Figure 2. The smart glasses with camera.

There are various standards for camera connectivity that are supported with A Raspberry Pi board such as SPI, I2C, CSI1, CSI3, Usb Serial, etc. In this work, we use Usb Serial camera. Its small size is suitable for a wearable device. Furthermore, it is also supported with a Usb OTG adapter. We also select the cameras that support MJPEG standard. Such devices allow built-in data compression, therefore the amount of data in transmission and storage can be reduced. In our system, the delay for MJPEG data capture and transmitted to the Raspberry Pi board is less than 0.1 s. The camera consumes 4 mA in idle mode, and 10 mA in photo mode. The camera is set to capture photos after every 1 min , thus about 1440 photos are taken per day.

After tuning the system, the board consumes on average 66 mA (without WiFi) and 106 mA (with WiFi). In total, the device (board and camera) consumes 70 mA in idle mode (without WiFi) and 116 mA in active mode (with WiFi). To reduce energy consumption, WiFi is only used for data synchronization, i.e. when it captures a new photo and sends this photo to the server. In our experiments, it takes 8 s for each WiFi connection session on Raspberry Pi Zero W for indoor environment. As the time interval between two continuously taken photos is set to be 1 min, we can estimate the average energy to be consumed in each cycle (for one photo) as follows: 52 s $\times$ 70 mA + 8 s $\times$ 116 mA = 4568 mAs.

The device is supplied by a PowerBank with 5000 mAh (= 18,000,000 mAs). The power supply can be used for 18,000,000/4568 = 3940 cycles, which means the device can be running for up to 65.5 h (more than 2.5 days).

2.2.2. Connection and Server

The transmission of images from the smart glasses to our server is conducted using MQTT. MQTT is a standard for lightweight publish and subscribe systems, especially for constrained devices and with low- bandwidth as the device in our work. Using MQTT, a device can send and receive messages as a client. We handle the processing tasks in our server using Nodejs. We also use Node-RED, an open source solution developed by IBM, to link the operations between devices and the server.

MQTT is a stable transmission method. However, in the case users lose their connection to the server, we also use local storage to keep the photos, and then upload to the server when connection is reestablished. The capacity of the local storage is about 16 GB, divided into 4 GB for operating system and 12 GB for data storage. An image taken from our device consumes about 130–220 KB. Thus, our device can keep over 90,000 photos.

2.3. Our Image Captioning Method

2.3.1. Model Overview

Our model in this work is based on our previous model [17]. Our model follows the encoder–decoder framework, which consists of an encoder to extract features from the images and a decoder to generate captions from the extracted features. Our encoder extracts tags features and regional features from the image, and then processes tag features and regional features with attention mechanism to produce the final combined features. Our decoder produces image captions from the combination of the two features by generating one word at each time step.

Figure 3 shows the overview of our model. From a given image, tags and regional features are extracted by two models (YOLO9000 and Faster R-CNN, respectively). Each kind of features is processed through an attention module to produce local features that represent the part the model is currently focused on. The two local features are combined and fed into an LSTM model [13] to generate the probabilities of the words in the vocabulary set at each time step. The generated results are processed with a beam search strategy to choose the best candidate caption.

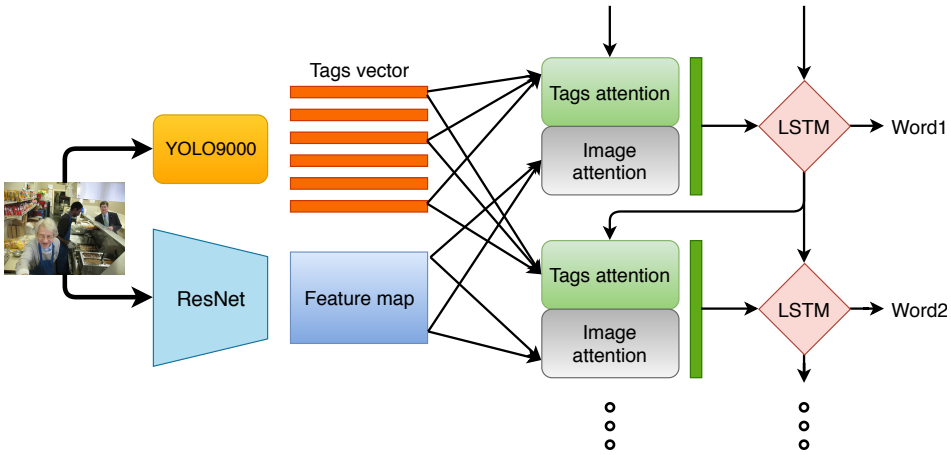

Figure 3. Model overview.

2.3.2. Extracting Image Features

The image features extracting module in our model is similar to our previous work [17]. We use a convolutional neural network to extract the features from the images. We choose ResNet [18] to be

our feature extracting module. The ResNet model [18] uses residual blocks in its structure to create shortcuts between layers. This helps the model keep information through layers of the network, resulting better performance compared to VGG [19].

Given an input image, we first resize the image into the shape of 224 × 224. The image is then process through a pretrained model of ResNet to produce the image features. We extract the feature map from the last convolutional layer of the ResNet model with the shape of 14 × 14 × 1024. The feature map is then fed into the image attention module.

2.3.3. Extracting Tags with YOLO9000

We extracted the tags feature using YOLO9000 model [20]. The YOLO9000 model is based on the previous YOLO model [6]. Joseph Redmon et al. [20] proposed various improvements to the YOLO model and then jointly trained it on both classification dataset and detection dataset to create YOLO9000 model. Due to the jointly training, the model is able to predict detections for objects that do not have labeled data in the detection dataset. The YOLO9000 model also shows better performance in both speed and accuracy compared to some other methods such as Faster R-CNN [21] and SSD [22] (Table 1). In our previous work [17], we used a Mask R-CNN model [5] trained on MSCOCO dataset [23] as our tags detector. The Mask R-CNN [5] can only detect up to 80 different tags of the MSCOCO dataset while the YOLO9000 can detect over 9000 different tags. This motivates us to replace the Mask R-CNN model with the YOLO9000 model. The increased number of different tags can help the model deal with concepts not in 80 different concepts of MSCOCO dataset and also enhance the information for the captions generating module.

From a given image as input, we first extract a list of tags using YOLO9000. We only keep top 20 tags with highest probabilities. We then break each tag into words if the tag contains more than one word. Redundant words are eliminated so that the list will contain only unique words. After this process, an image will produce at most 23 words. Therefore, we choose the maximum size of our list to be 23. If the list has fewer than 23 words, we pad special <NULL> tokens into the list to keep all the lists at the same size.

Table 1. Comparison between YOLO9000 and other methods. The higher is the mAP score, the better (source from [20]).

Detection Frameworks	mAP	FPS
Faster R-CNN VGG-16 [21]	73.2	7
Faster R-CNN ResNet [21]	76.4	5
SSD300 [22]	74.3	46
SSD500 [22]	76.8	19
YOLO9000 [20] 480 × 480	77.8	59
YOLO9000 [20] 544 × 544	78.6	40

Each word i, including <NULL> token, is represented by a one-hot vector V_i of N dimensions (N is the size of the vocabulary set). We then embed each word into d-dimension space using word embedding method [24]. Concretely, we use an embedding matrix E and compute the dot product v_i between the one-hot representation V_i of a word and the embedding matrix E as in Equation (1).

$$v_i = V_i \cdot E \tag{1}$$

The embedding matrix is initially assigned with random value and then trained along with the whole model. The purpose of using embedding matrix is to transfer representation of a word into a more meaningful space that can represent the semantic relationship between words. After the embedding process, a list of words from the image will be represented by a list of d-dimension

embedded vectors. This tags features will bed fed into the tag attention module to produce the feature vector for generating captions.

2.3.4. Attention Modules

In our work, we use two different attention modules, one for tags feature and one for image features. We based our attention mechanism on the method in work of Kelvin Xu et al. [14].

In the image attention module, given the feature map with shape of $14 \times 14 \times 1024$ extracted from ResNet model [18], we assign a weight value α within the interval (0,1) to each of $14 \times 14 = 196$ regions. The value of α represents how much the model is paying attention on that corresponding region. If α is high (closer to 1), the information of that region is kept. Otherwise (closer to 0), the information of that region is suppressed. The value of α is computed from the information from the previous hidden state combined with the feature of the corresponding region (as in Equation (2)).

$$\alpha_{ti} = f_{softmax}(f_{attend1}(a_i, h_{t-1})) \tag{2}$$

where α_{ti} is the weight value of region i at time step t. a_i is the 1024-dimension feature vector of the region i. h_{t-1} is the hidden state from the previous time step. The $f_{attend1}$ is in fact a small network that is trained along with our whole model to learn to adjust the weights α automatically based on training data. The weights α is then used to produce the context vector $z1$ of the image as follows:

$$z1_t = \sum_{i=1}^{N \times N} \alpha_{ti} \cdot a_i \tag{3}$$

The region with α closer to 1 contributes more to the final context vector. Because α is changed for each time step, the context vector is also changed, resulting in the model focusing on different regions while generating the caption.

We also apply this mechanism in the tags attention module. Given a set of K 512-dimension tag vectors, the tags attention module computes a set of weight values β that represents the attention the model pays on each tag. Each value β_i assigned for the tag vector i is computed as follows:

$$\beta_{ti} = f_{softmax}(f_{attend2}(v_i, h_{t-1})) \tag{4}$$

Similar to Equation (2), h_{t-1} is the hidden state from the previous time step. $f_{attend2}$ is a network that is trained on the data along with the whole model. v_i is the tag vector i in the list. We then compute the second context vector $z2$ using the weight values β.

$$z2_t = \sum_{j=1}^{K} \beta_j \cdot v_j \tag{5}$$

The two context vector along with the embedded vector of the word generated from previous times step are then combined via concatenating to produce the final context vector z. The final context vector z is then fed into the LSTM model for generating captions.

2.3.5. Beam Search Strategy

The LSTM model [13] takes the context vector z from the attention module and computes the hidden state h_t at each time step t, as follows:

$$f_t = \sigma_g(W_f z_t + U_f h_{t-1} + b_f) \tag{6}$$

$$i_t = \sigma_g(W_i z_t + U_i h_{t-1} + b_i) \tag{7}$$

$$o_t = \sigma_g(W_o z_t + U_o h_{t-1} + b_o) \tag{8}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c z_t + U_c h_{t-1} + b_c) \tag{9}$$

$$h_t = o_t \circ \sigma_h(c_t) \tag{10}$$

where f_t, i_t, o_t are forget gate, input gate and output gate, respectively. The forget gate f_t adjusts the information from the previous time step used to compute the new hidden state while input gate i_t adjusts the information from the input vector z_t. The output is the processed through the output gate o_t to compute the final hidden state h_t. The hidden state is then processed through a classifier to compute the probabilities of all words in the vocabulary set. The generated word is chosen based on the probability of each word. The hidden state h_t is also fed back into the attention modules to compute new context vector z_t. The process is repeated until the model generates a special <END> character that mark the end of the sentence.

During training, we use the word from ground truth to compute the context vector z_t in order to avoid error from the previous time step. The generated caption is compared with the ground truth to compute the loss function. We train our whole model by minimizing the loss function stochastic gradient descent with momentum.

During testing, the ground truth is not accessible. Therefore, the previous generated word is used to compute the context vector z. This could lead to a chain of errors if the previous word is wrongly selected. To avoid this, we apply our modified beam search strategy presented in [17]. Figure 4 shows our beam search strategy. Given probabilities of all words in the vocabulary set, we choose top k words with the highest probabilities, excluding the special <END> character. Then, each of the k words is used to generate a new list of probabilities. The new generated probabilities is combined with the probability of their ascendants to compute the scores of the sequences as follows:

$$score(w) = \log P(w) + \log P(Ascendants(w)) \tag{11}$$

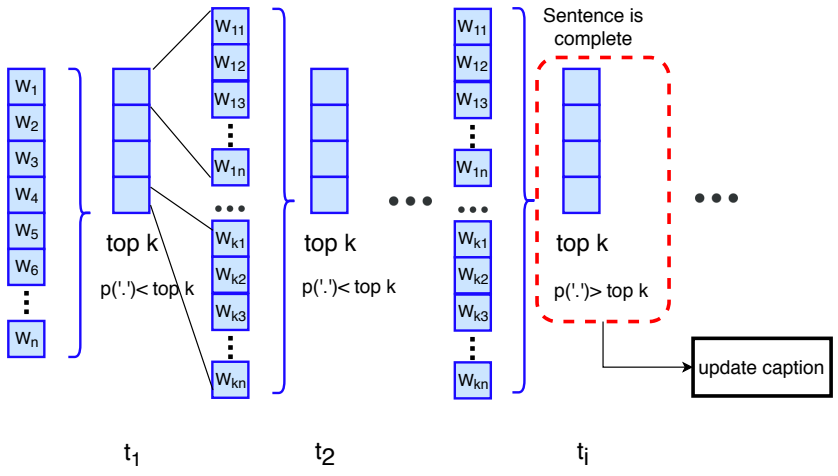

Figure 4. Our beam search strategy.

From the scores of the sequences, we choose the next top k for the next time step. Unlike beam search strategy in some other methods, beside choosing the sequence with the highest score to be

the final candidate, we apply a heuristic to adjust the result. Choosing the sequence with the highest score may lead to an incomplete sentence because longer sentences get lower scores due to more multiplication. To avoid this, we use the probability of the <END> character to select the final candidate. If the probability of the <END> character is in the top k, we consider that sequence could be a complete sentence. If the current candidate is an incomplete sentence, it is immediately replaced by the new sequence that has high probability of the <END> character. If both are complete sentence, the scores are compared to choose the better candidate. With this heuristic, our model favors complete sentence over incomplete ones. Additional our previous work [17], in this work, we add one more heuristic to favor long sentences over short sentences because longer sentence can describe more details than a shorter one. We apply the formula in [25] to adjust the score at each time step. Concretely, we divide the score with the length normalization score so that the scores of long sentences do not reduce rapidly due to the multiplication of small numbers. The length normalization is computed as follows:

$$normalize(w) = score(w)/lp(w) \tag{12}$$

$$lp(w) = \frac{(5+|w|)^{\gamma}}{(5+1)^{\gamma}} \tag{13}$$

where $score(w)$ is the probability of the sequence computed as in Equation (11). We conducted experiments on different beam size and γ to find the most suitable configuration for our model. The training and testing results are reported in the next section.

3. Results

In this section, we report our experiment on the MSCOCO dataset. The results were evaluated on four different metrics: BLEU [26], METEOR [27], ROUGE-L [28] and CIDEr [29]. We also report results from some other methods for comparison.

3.1. Datasets

We trained and tested our model on MSCOCO dataset version 2017 [23] (the MSCOCO dataset can be downloaded via this http://cocodataset.org). The dataset was divided into training set, validation set and testing set, which had about 118,000 images, 5000 images and 41,000 images, respectively. Each image was annotated with five different captions from different people using Amazon Mechanical Turk. The context of the images mainly focus on objects that belong to 80 common categories including animal, vehicle, food or kitchen furnitures.

3.2. Evaluation Metrics

We evaluated the results of our method using four metrics: BLEU [26], METEOR [27], ROUGE-L [28] and CIDEr [29]. These metrics are used to compute the similarity between two sentences. They are first introduced in the domain of machine translation and brought into image captioning to measure the similarity of the generated captions and the ground truth. Each metric computes the similarity in a different way. BLEU [26] uses the n-gram precision to calculate the similarity score. The higher n is, the more precise the generated captions need to be in order to get high scores. We report the BLEU metric with n from 1 to 4. METEOR [27] calculates the score using matching method. Synonym and stem are also considered in the matching method. ROUGE-L [28] calculates based on the longest common subsequence (LCS) between the generated captions and the ground truth. CIDEr metric uses n-gram combined with TF-IDF to represent a sentence as a vector and then computes the cosine similarity between the vectors.

We use these metrics to measure the similarity between the generated captions and the ground truth in the dataset. The implementation of these metrics are also provided along with the MSCOCO dataset. For comparison, we use the same implementation of these metrics as in some other methods.

3.3. Experimental Results

We trained our model on MSCOCO dataset on a computer with support of GPU NVIDIA Tesla K80 11 GB. We combined the 1024-dimension feature vector from the image attention module, the 512-dimension tag vector from the tags attention module and the 512-dimension embedded vector of the previously generated word to create the 2048-dimension context vector. The context vector was then fed into LSTM model to generate one word at each time step. We optimized the loss function to train the LSTM model and the two attention modules. We chose batch size to be 64. The procedure took about 40 h. We also used early stopping with patient value of 20 to find the best set of parameters.

We tried different beam size with the above training configuration to find the best beam size in our method. The results are shown in Table 2. As shown in Table 2, the scores may vary with different beam size but models with beam size in of 2–4 showed highest scores on different metrics. Therefore, we chose beam size for our model to be 3, which showed highest scores on most metrics.

Table 2. Results our model on different beam size.

Beam Size	B-1	B-2	B-3	B-4	METEOR	Rouge-L	CIDEr
1	0.713	0.542	0.397	0.291	0.245	0.525	0.934
2	**0.727**	**0.560**	0.420	0.316	0.252	0.536	0.984
3	**0.727**	**0.560**	0.422	0.321	**0.254**	**0.538**	0.989
4	0.725	0.559	**0.423**	**0.322**	0.253	0.537	**0.990**
5	0.724	0.557	0.422	**0.322**	0.253	0.536	0.985
6	0.722	0.556	0.421	**0.322**	0.252	0.535	0.984
7	0.721	0.555	0.420	0.320	0.252	0.534	0.983
8	0.720	0.555	0.420	0.321	0.252	0.534	0.981
9	0.720	0.554	0.419	0.320	0.252	0.534	0.980
10	0.720	0.554	0.419	0.319	0.252	0.533	0.980

With beam size set to 3, we tried experiments on different γ value for length normalization. We tried value of γ from 0 to 1 with step of 0.1. The results are shown in Table 3. The scores of different γ values were quite similar. The lowest on BLEU-1 metric was 0.726 with γ = 0.9 and the highest was 0.728 with γ = 0.4 and 0.5. We chose γ for our model to be 0.4 because it had the most highest scores in all evaluation metrics.

Table 3. Results of our model on different length normalization γ.

Length Norm	B-1	B-2	B-3	B-4	METEOR	Rouge-L	CIDEr
0.0	0.727	0.560	0.422	0.321	0.254	0.538	0.989
0.1	0.727	0.560	0.422	0.321	0.254	0.539	0.991
0.2	0.727	0.560	0.423	0.321	0.254	0.539	0.991
0.3	0.727	0.560	0.423	0.322	0.254	0.539	0.992
0.4	**0.728**	**0.562**	**0.424**	**0.323**	**0.254**	**0.539**	**0.995**
0.5	0.728	0.561	0.424	0.322	0.254	0.539	0.993
0.6	0.727	0.560	0.423	0.321	0.254	0.538	0.990
0.7	0.727	0.561	0.423	0.322	0.254	0.539	0.992
0.8	0.727	0.560	0.423	0.322	0.254	0.538	0.992
0.9	0.726	0.560	0.422	0.320	0.253	0.538	0.990

We set the beam size to be 3 and the γ value to be 4 in our final model. We then tested our model on MSCOCO test server. Table 4 shows result of our model on MSCOCO test set with fvie captions per image. We also report results from other methods and result from our previous model for comparison. More discussion about the results are presented in Section 4.

Table 4. Results of our models compared to other methods.

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L	CIDEr
Watson Multimodal [30]	0.781	0.619	0.470	0.352	0.270	0.563	1.147
Postech_CV [31]	0.743	0.575	0.431	0.321	0.255	0.539	0.987
ATT [15]	0.731	0.565	0.424	0.316	0.250	0.535	0.943
Google-NIC [11]	0.713	0.542	0.407	0.309	0.254	0.530	0.943
Montreal/Toronto [14]	0.707	0.492	0.344	0.243	0.239	-	-
Nearest Neighbor [32]	0.697	0.521	0.382	0.28	0.237	0.507	0.886
NeuralTalk [12]	0.650	0.464	0.321	0.224	0.21	0.475	0.674
80_tags_model [17]	0.701	0.527	0.384	0.277	0.230	0.511	0.835
9000_tags_model (ours)	**0.723**	**0.554**	**0.414**	**0.310**	**0.250**	**0.532**	**0.942**

We also visualized the results of our captioning method. Figure 5 shows some example captions that were generated by our captioning method. The images were taken from the COCO test set, which means that the model did not see the image during training. The generated captions were relevant to their corresponding image. The captions also described the content using clear and precise words such as "dog", "cat", "banana", and "park" instead of general words such as "animal" or "fruit". This made retrieval based on the captions more accurate. However, our model still made some mistakes. For example, in the third image, it hallucinated a table while there was no table in the images. The model may have assume that a bowl is often put on a table. This requires more research to be done to solve this problem.

Figure 5. Some generate captions of our model from MSCOCO test set.

We also tried our model on real life images collected by ourselves. The results illustrated in Figure 6 show that our model could do well on real life images. Keywords such as beach were used in the description. When a user wants to retrieve when he/she went to a beach, a simple query with the keyword "beach" would help retrieve such events easily. Our model could also reason where to look while generating the description, as shown in Figure 7.

We used our trained model in our system. It took about 2 s to process an input image and our model could process 64 images simultaneously in one forward running. In real time running, the server would take 2 s to process the image taken from camera every minute.

Figure 6. Some generate captions of our model from images taken by ourselves.

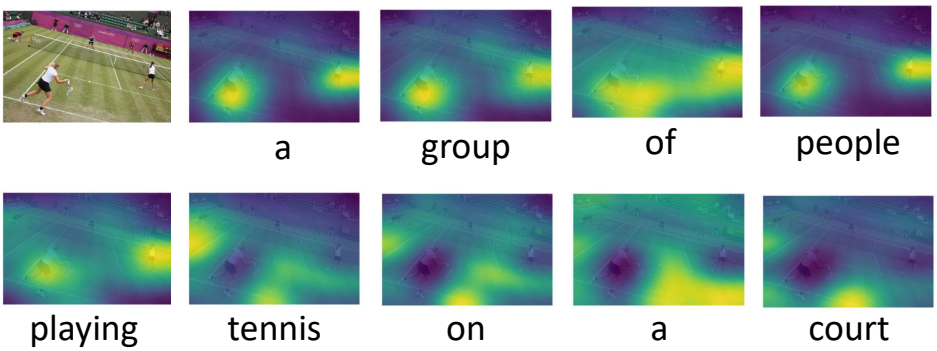

Figure 7. Visualization of our model's ability to focus on certain regions while generating the caption.

4. Discussion

In this section, we discuss more details about the result of our method. As shown in Table 4, our replacement of the tags extracting module with YOLO9000 showed great improvement. Compared to our previous method [17], we achieved better results on all evaluation metrics. On BLEU metric, we obtained about 0.03 higher score on average. On METEOR and ROUGE-L metric, we obtained about 0.02 higher scores. Especially on CIDEr metric, we achieved a significant improvement with 0.11 higher value, an increase from 0.835 to 0.942. This shows that, with more detected tag names, the model receives more information, therefore can generate more accurate captions. We compared the results with other methods. We achieved higher results on all metrics compared to some previous method such as NeuralTalk [12], Nearest Neighbor [32], and Montreal/Toronto [14], and slightly higher compared to Google-NIC [11]. However, our work was still lower than some current state-of-the-art methods such as Watson Multimodal [30]. Despite not achieving high scores in the evaluation metrics, our model could generate captions with clear and rich information that is helpful for retrieval. As shown in Figure 5, the generated captions were relevant to the main content of the image and describe enough detail such as the name of the fruits or the location in the image. These information can be used to easily retrieve relevant content given a query. Our model can also focus on certain regions to generate the corresponding words. This let us keep track of the generating process in case of wrong caption is generated.

We developed our system for generating personal lifelog caption using our proposed image captioning method. From photos take from smart glasses, our system can generate description relevant to the content. When a user is staying in the smart environment where connection to the server is available, such as a smart home or smart office, photos can be transmitted periodically (1 photo/minute) to the server, and then our model will process the image and add the description of that photo to the database within 2 s. This allows real time capturing of important events for the user. When the user moves out of the smart environment and loses connection, photos can be stored in a SD-card of the smart glasses and will be uploaded to our server later. The design and implementation of smart glasses are also optimized for more effective transmission and energy consumption.

For future research, we aim to extend the proposed method with more domain knowledge to further improve the quality of image description. Besides, security features should be considered. Some security methods should be integrated into the system of personal lifelog generation to protect communications between devices and the server. We also intend to apply document embeddings into our system for better querying. Users can query the events of interest based on semantic similarity of the needs and the captions.

5. Conclusions

In this article, we propose two adjustments to our previous method for image captioning. First, we use a new objects detector, YOLO9000, as tags extracting module to enhance the tags features. Second, we apply a new normalization method to the beam search strategy in order to make the model favor longer captions. These adjustments work as intended and show improvement in our results. Although our system still needs more improvement, it does show great potential application in capturing personal lifelog in real time.

Author Contributions: Conceptualization, Q.-A.L., V.-K.V.-H. and M.-T.T.; methodology, V.-K.V.H. and D.-T.N.; software, Q.-A.L. and V.-K.V.-H.; hardware, D.-T.N. and M.-K.T.; validation, Q.-A.L., V.-K.V.-H., D.-T.N. and M.-K.T.; formal analysis, Q.-A.L. and V.-K.V.-H.; investigation, V.-K.V.-H., D.-T.N. and M.-K.T.; resources, D.-T.N. and M.-K.T.; data curation, D.-T.N. and M.-K.T.; writing—original draft preparation, Q.-A.L.; writing—review and editing, Q.-A.L., V.-K.V.-H. and M.-T.T.; visualization, V.-K.V.-H.; supervision, M.-T.T.; project administration, M.-T.T.; and funding acquisition, M.-T.T.

Acknowledgments: We would like to thank AIOZ Pte Ltd for supporting our research team in this project.

Conflicts of Interest: The author declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MDPI	Multidisciplinary Digital Publishing Institute
LSTM	Long short-term memory
TF-IDF	Term Frequency-Inverse Document Frequency
IoT	Internet of Things
SoICT	Symposium on Information and Communication Technology
MS	Microsoft
COCO	Common Objects in COntext
CLEF	Conference and Labs of the Evaluation Forum

References

1. Nguyen, V.T.; Le, K.D.; Tran, M.T.; Fjeld, M. NowAndThen: A Social Network-based Photo Recommendation Tool Supporting Reminiscence. In Proceedings of the 15th International Conference on Mobile and Ubiquitous Multimedia, Rovaniemi, Finland, 12–15 December 2016; ACM: New York, NY, USA, 2016; pp. 159–168.
2. Dang-Nguyen, D.T.; Piras, L.; Riegler, M.; Zhou, L.; Lux, M.; Gurrin, C. Overview of ImageCLEFlifelog 2018: Daily Living Understanding and Lifelog Moment Retrieval. In Proceedings of the 5th Italian Workshop on Artificial Intelligence and Robotics (AIRO 2018), Trento, Italy, 22–23 November 2018.
3. Gurrin, C.; Smeaton, A.F.; Doherty, A.R. LifeLogging: Personal Big Data. *Found. Trends Inf. Retr.* **2014**, *8*, 1–125. [CrossRef]
4. Rawassizadeh, R.; Tomitsch, M.; Wac, K.; Tjoa, A.M. UbiqLog: A generic mobile phone-based life-log framework. *Pers. Ubiquitous Comput.* **2013**, *17*, 621–637. [CrossRef]
5. He, K.; Gkioxari, G.; Dollar, P.; Girshick, R. Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017.
6. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.

7. Minh-Triet, T.; Thanh-Dat, T.; Tung, D.D.; Viet-Khoa, V.H.; Quoc-An, L.; Vinh-Tiep, N. Lifelog Moment Retrieval with Visual Concept Fusion and Text-based Query Expansion. In Proceedings of the CLEF 2018 Working Notes, Avignon, France, 10–14 September 2018.
8. Farhadi, A.; Hejrati, M.; Sadeghi, M.A.; Young, P.; Rashtchian, C.; Hockenmaier, J.; Forsyth, D. Every Picture Tells a Story: Generating Sentences from Images. In Proceedings of the 11th European Conference on Computer Vision: Part IV, Crete, Greece, 5–11 September 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 15–29.
9. Kulkarni, G.; Premraj, V.; Dhar, S.; Li, S.; Choi, Y.; Berg, A.C.; Berg, T.L. Baby Talk: Understanding and Generating Simple Image Descriptions. In Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition, Colorado Springs, CO, USA, 20–25 June 2011; IEEE Computer Society: Washington, DC, USA, 2011; pp. 1601–1608.
10. Yatskar, M.; Galley, M.; Vanderwende, L.; Zettlemoyer, L. See No Evil, Say No Evil: Description Generation from Densely Labeled Images. In Proceedings of the Third Joint Conference on Lexical and Computational Semantics (*SEM 2014), 2014, Dublin, Ireland, 23–24 August 2014; pp. 110–120.
11. Vinyals, O.; Toshev, A.; Bengio, S.; Erhan, D. Show and Tell: A Neural Image Caption Generator. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015.
12. Karpathy, A.; Fei-Fei, L. Deep Visual-Semantic Alignments for Generating Image Descriptions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015.
13. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
14. Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; Bengio, Y. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015; Bach, F.; Blei, D., Eds.; PMLR: Lille, France, 2015; Volume 37, pp. 2048–2057.
15. You, Q.; Jin, H.; Wang, Z.; Fang, C.; Luo, J. Image Captioning With Semantic Attention. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
16. Anderson, P.; He, X.; Buehler, C.; Teney, D.; Johnson, M.; Gould, S.; Zhang, L. Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018.
17. Viet-Khoa, V.H.; Quoc-An, L.; Duy-Tam, N.; Mai-Khiem, T.; Minh-Triet, T. Personal Diary Generation from Wearable Cameras with Concept Augmented Image Captioning and Wide Trail Strategy. In Proceedings of the Ninth International Symposium on Information and CommunicationTechnology (SoICT 2018), DaNang City, Viet Nam, 6–7 December 2018.
18. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
19. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
20. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. *arXiv* **2016**, arXiv:1612.08242.
21. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems (NIPS)*; Curran Associates, Inc.: Palais des Congrès de Montréal, Montréal, Canada, 2015.
22. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In Proceedings of the ECCV, Amsterdam, The Netherlands, 11–14 October 2016.
23. Lin, T.; Maire, M.; Belongie, S.J.; Bourdev, L.D.; Girshick, R.B.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. *arXiv* **2014**, arXiv:1405.0312
24. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, arXiv:1301.3781.
25. Wu, Y.; Schuster, M.; Chen, Z.; Le, Q.V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv* **2016**, arXiv:1609.08144.

26. Papineni, K.; Roukos, S.; Ward, T.; Zhu, W.J. BLEU: A Method for Automatic Evaluation of Machine Translation. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics; Association for Computational Linguistics, Stroudsburg, PA, USA, 7–12 July 2002; pp. 311–318. [CrossRef]
27. Lavie, A.; Agarwal, A. Meteor: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. In Proceedings of the Second Workshop on Statistical Machine Translation, Prague, Czech Republic, 23 June 2007; Association for Computational Linguistics: Stroudsburg, PA, USA, 2007; pp. 228–231.
28. Lin, C.Y. ROUGE: A Package for Automatic Evaluation of Summaries. In Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004), Barcelona, Spain, 25–26 July 2004.
29. Vedantam, R.; Lawrence Zitnick, C.; Parikh, D. CIDEr: Consensus-Based Image Description Evaluation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015.
30. Rennie, S.J.; Marcheret, E.; Mroueh, Y.; Ross, J.; Goel, V. Self-critical Sequence Training for Image Captioning. *arXiv* **2016**, arXiv:1612.00563.
31. Mun, J.; Cho, M.; Han, B. Text-guided Attention Model for Image Captioning. *arXiv* **2016**, arXiv:1612.03557.
32. Devlin, J.; Gupta, S.; Girshick, R.B.; Mitchell, M.; Zitnick, C.L. Exploring Nearest Neighbor Approaches for Image Captioning. *arXiv* **2015**, arXiv:1505.04467.

Sample Availability: The source code of the model from the authors is available in the following https://github.com/vhvkhoa/image-captioning-with-concept.