Meaning guided video captioning

Rushi J. Babariya[1], Toru Tamaki[2]

[1]BITS Pilani, India, [2]Hiroshima University, Japan

Abstract. Current video captioning approaches often suffer from problems of missing objects in the video to be described, while generating captions semantically similar with ground truth sentences. In this paper, we propose a new approach to video captioning that can describe objects detected by object detection, and generate captions having similar meaning with correct captions. Our model relies on S2VT, a sequence-to-sequence model for video captioning. Given a sequence of video frames, the encoding RNN takes a frame as well as detected objects in the frame in order to incorporate the information of the objects in the scene. The following decoding RNN outputs are then fed into an attention layer and then to a decoder for generating captions. The caption is compared with the ground truth by learning metric so that vectors of generated captions are semantically similar to those of ground truth. Experimental results with the MSDV dataset demonstrate that the performance of the proposed approach is much better than the model without the proposed meaning-guided framework, showing the effectiveness of the proposed model.

Keywords: video captioning, sequence-to-sequence, object detection, sentence embedding

1 Introduction

The task of describing a video with a text has been receiving a great attention in recent years. The mapping from a sequence of frames to a sequence of words was first introduced with a sequence-to-sequence model [1], then a variety of models [2] have been proposed. However, these approaches often suffer from some common problems. First, captions should reflect objects in the scene while generated captions may not include terms indicating such objects. This issue is caused by captioning models that takes frames for capturing features for generating captions, not for detecting objects in the scene. Second, generated captions are evaluated with ground truth captions by using loss functions, which typically compare two sentences in a word-by-word manner. This may not reflect a semantic similarity between sentences because the change of a single word in a sentence could lead to a completely opposite meaning, but the loss might be small due to the difference of the single word.

In this paper, we propose a meaning-guided video captioning model in cooperating with an object detection module. Our model uses encoder LSTMs to learn the mapping from a video to a description, of which back born network

is the a sequence-to-sequence video-to-text model, called S2VT [1]. Upon this base network, we feed object detection results [3] of each frame into the encoder LSTMs to extract the most dominant object in each frame. Our model further incorporates attention in decoding LSTMs for enhancing information of frames that characterize the given video. In addition, we proposes a new approach to train the proposed video-to-text model. Instead of a classical training using a word-by-word loss, we train the model to learn the meaning of captions, or semantic similarity of captions. To this end, we propose a metric leaning model to embed captions so that distances between a semantically similar pair of captions becomes smaller than a dissimilar pair.

2 Related work

There are many works on video captioning. The early model used a sequence-to-sequence model (S2VT) [1]. This was inspired by a sequence-to-sequence translation model that takes a text in one language and output a text in another language. Instead, the S2VT model takes a sequence of video frames as input to encoder LSTMs, and outputs a sequence of words through decoder LSTMS. Later a 3DCNN was used to extract video features [2] to generate texts describing videos, and also attention has been used [4] to find which part of the video are more informative.

Image captioning [5,6,7] is a closely related task that describing image, instead of videos. Some works for image captioning have aware the issue that generated captions may miss object in the scene [8], however not well studied for the video captioning task. This can be supported by the help of object detection [3]. We therefore use object detectors to find objects in the scene and then reflect the object information in generated captions.

Designing the loss function is a key for many models to success, and for video captioning we need a loss to compare generated and ground truth captions. This is common for many text-related tasks such as image and video captioning and visual question generation (VGQ) [9]. A problem is that a loss usually compares texts word-by-word, which is fragile to a little difference of words in sentences. Furthermore, a typical dataset for captioning has several different captions as ground truth of a single video, which is another cause for the word-by-word loss to be confused. In the proposed model, we propose a loss using sentence embedding and metric leaning so that semantically similar captions have small distances while different captions are large apart from each other.

3 Encoder-decoder model

Our proposed model is built on top of a baseline sequence-to-sequence video-to-text generator, S2VT [1]. The baseline model uses a stacked 2-layer LSTM encoder-decoder model that takes a sequence of RGB frames $f_1, f_2, \ldots, f_N$ as input and produces a caption or a sequence of words $w_1, w_2, \ldots, w_M$. Frame features are extracted by using the VGG16 pre-trained model. The lower LSTM

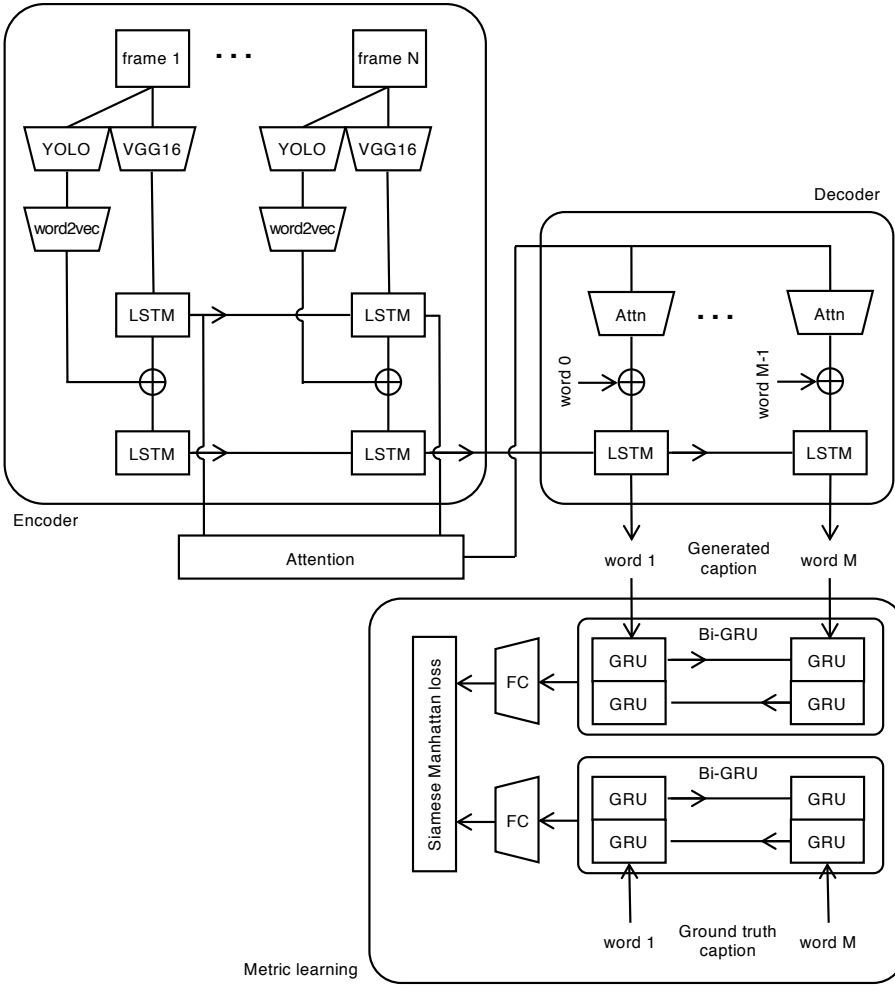

Fig. 1. Our model

in the encoder takes the output of the upper LSTM, encoding the visual information, concatenated with padding due to the absence of text information. In contrast, in the decoder the upper LSTM is fed padding due to the lack of video frames, and the lower LSTM takes the concatenation of padding and word information.

The proposed model is shown in Figure 1. It consists of encoder, decoder, and metric learning components.

3.1 Encoder

The encoder LSMT now takes the concatenation of holistic visual information and scene object information (instead of padding). VGG16 features of 2048 dimension is extracted from a current RGB frame as holistic visual information. However it would not reflect objects appear in frames, and therefore we use the YOLOv3 object detector [3]. It may find many objects in a frame, however we focus on the dominant object in each frame. Specifically, we pick up the object having the highest objectness score in the YOLO detector, and find the string describing the category of the object (e.g., 'person' or 'cat'). The string is embedded with word2vec [10,11] to convert it to an embedding vector of 300 dimension.

The upper LSTM in the encoder takes the 2048-d visual vector of the frame, then the hidden state of 1000 dimension is passed to the lower LSTM after concatenating with the object embedding vector, resulting in a 1300-d vector to be fed to the lower LSTM.

The lower LSTM outputs a 1000-d hidden state vector passing through the encoder, then to the decoder LSTM. In contrast, 1000-d hidden states of the upper LSTM are not passed to the decoder, but to the attention layer.

3.2 Attention

Let $h_1, \ldots, h_N$ be the hidden states of the upper encoder LSTM. These are stacked in column-wise to make a matrix

$$H = (h_1, \ldots, h_N) \quad \in R^{1000 \times N}, \tag{1}$$

where N is the number of video frames encoded. This is used as attention [] for $s_t \in R^{1000}$, a given output of the decoder LSTM at time step t for $t = 1, \ldots, M$. To do so, we construct an activation energy vector of the following form [12]

$$\lambda_t = \mathrm{softmax}(s_t^T W H), \tag{2}$$

where $W \in R^{1000 \times 1000}$ is a trainable linear layer. Using λ_t, we have the attention vector of 1000 dimension as $a_t = H\lambda_t$.

This attention vector is used as input at time step t to the decoder LSTM after a linear layer keeping dimension and concatenation with the word embedding w_t of 300 dimension at time t.

3.3 Decoder

The decoder LSTM takes the 1000-d hidden state from the lower decoder LSTM, and the input (the concat of attention a_t and word embedding w_t). The output is 1000 dimension and fed into a linear layer to convert a vector of vocabulary size of 25231 words (in the case of the experiments below). This is then passed to a softmax layer to obtain the word probability p_t at time t.

3.4 Word-by-word loss

This probability p_t is used to compute the cross entropy with the word w_t in the ground truth caption. The sum of this word-wise cross entropy for $t = 1, \ldots, M$ is used as a loss to train the network.

This loss has been typically used for train networks to compare generated and ground truth captions in the literature [13,9]. However, it compares texts word-by-word, which is fragile against a little difference of words in sentences. Furthermore, a typical dataset for captioning has several different captions as ground truth of a single video, which is another cause for this word-by-word loss to be confused.

Therefore, in the training procedure, we first use this loss to train the network until convergence, then switch to another loss that captures the semantic similarity between captions, which is described next.

4 Metric learning component for captions

4.1 Soft-embedding sequence generation

In order to construct a loss comparing generated and ground truth captions, our model generates captions during training. To this end, a possible way might be sampling the next word by using the word probability p_t. This is however not useful for training because the sampling procedure cuts the computation graph and back propagation doesn't go back through the decoder LSTM.

Instead, we propose to use the probability p_t as weights for the next word. If it was a one-hot vector, then finding the next word is simply picking up the corresponding column of the 300×25231 word embedding matrix E, or equivalently multiplying the one-hot vector to E. As the similar way, we construct a single word embedding by $s_t = Ep_t$. This is actually not any of words in the vocabulary, but should reflect a "soft" word choice of the decoder LSTM.

This is passed to the attention in the next time step, then the next weighted embedding word s_{t+1} is computed. Eventually, the decoder LSTM outputs the sequence $s_1, \ldots, s_M$ as a generated caption for the given video.

4.2 Meaning-guided loss

Now we have two sequences; generated and ground truth captions. In our metric learning component, these captions are first embedded with a sentence-to-vector

model. This is a bi-directional GRU [14] with 1000-d hidden states each, resulting in 2000-d output. Then a linear layer is used to reduce the dimension to 1000.

To compare two 1000-d vectors v_1 and v_2, corresponding to generated and ground truth captions, we use the the Siamese Manhattan loss [15]

$$L_{\text{sim}}(v_1, v_2) = 1 - \exp(\|v_1 - v_2\|_1). \tag{3}$$

Now this loss should be small for v_1 and v_2 because these two captions should be similar and the vectors close to each other. In contrast, if we have two different captions v_3 and v_4, the situation is opposite and the following loss should be minimized.

$$L_{\text{dis}}(v_3, v_4) = \exp(\|v_3 - v_4\|_1). \tag{4}$$

The overall loss for this is given by

$$\begin{aligned} L = & E_{v_1, v_2 \sim \text{training sample pair}}[L_{\text{sim}}(v_1, v_2)] + \\ & E_{v_3, v_4 \sim \text{dissimilar sentence pair}}[L_{\text{dis}}(v_3, v_4)] \end{aligned} \tag{5}$$

4.3 The biased Judge

Here the Metric learning component(Bidirectional GRU) scores the semantic similarity of the generated caption by mapping the sentences to a common latent space , so it acts like a judge to the caption model. But in our model we train the judge(scorer) as well !!!!. Given a generated caption v_1 and ground truth caption v_2 ,the Judge assumes that the caption model is correct and v_1 should be semantically similar to v_2. So it corrects its judgment by back propagating the loss $L_{\text{sim}}(v_1, v_2)$ to the Metric , however the caption model might be giving a very different caption from the ground truth, and if this is the case then the judge might learn an identity mapping where it says all statements as Semantically similar. To prevent this from happening along with v_1 and v_2 we also provide two semantically different captions to the judge v_3 and v_4 , the judge now also back propagates $L_{\text{dis}}(v_3, v_4)$, and this prevents the judge(Metric) from becoming biased. Of course we need a pretrained caption model and a pretrained judge for this to work which is described later in the paper.

4.4 Intra-batch training

A possible drawback of the metric leaning component described above is that datasets for captioning do not provide any dissimilar sentence pairs. In the followings, we describe tricks to train the proposed model efficiently.

The first trick is to use a mini-batch for dissimilar pair sampling (Figure 2(top)). Suppose we are given a batch consists of 50 ground truth captions for 50 different videos in a dataset, and then we have corresponding 50 generated captions. Among these 100 captions, the 50 training pairs are used for L_{sim}. In addition, there are many more dissimilar caption pairs because different ground truth captions can be considered as different sentences. Therefore we can sampling a different caption pair in the batch for L_{dis}.

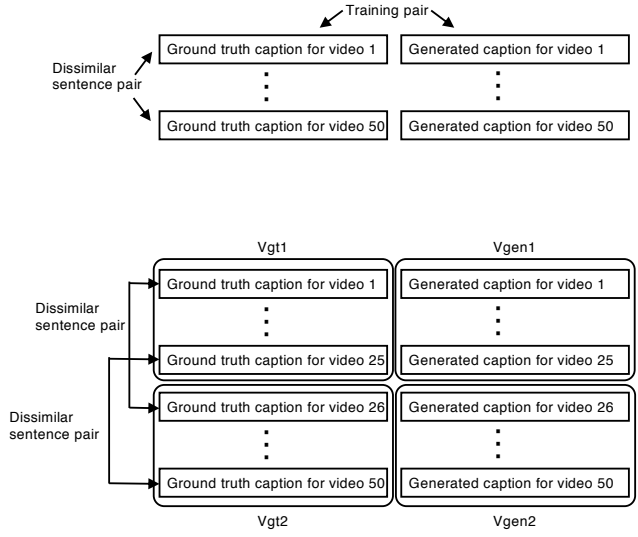

Fig. 2. Intra-batch sampling. (top) Sampling two ground truth sentences in a batch as dissimlar sentence pairs. (bottom) Use corresponding ground truth pairs as as dissimlar sentence pairs.

However, a naive sampling is inefficient because a ground truth caption is encoded in a vector several times; once as a training pair, and more as a dissimilar pair. This leads to encoding the same caption multiple times with the the network having the same weights. This is a waste of resource because the encoding results are the same before computing the loss and backprop.

Our second trick is to do this efficiently (Figure 2(bottom)). Again suppose we are given a 50-sample batch, and we have generated and ground truth caption embeddings as two matrices of size 50×1000 (each row is a 1000-d embedding vector). Let V_{gen} is the matrix of generated captions, and V_{gt} is the matrix of ground truth captions. We split them to two to obtain four 25×1000 matrices; $V_{\mathrm{gen1}}, V_{\mathrm{gt1}}$ and $V_{\mathrm{gen2}}, V_{\mathrm{gt2}}$. Now each row in V_{gen1} has nothing related to any row in V_{gen2} due to random sampling from the training dataset, and the same for $V_{\mathrm{gt1,2}}$. Therefore, we can use i-th rows of $V_{\mathrm{gen2}}, V_{\mathrm{gt2}}$ as the dissmilar pair for i-th rows of $V_{\mathrm{gen1}}, V_{\mathrm{gt1}}$, and vice versa. This can be done by keeping these embedding vectors just before computing the Siamese Manhattan loss.

Our third trick is to do it more efficiently. As a concept, i-th training pair and i-th dissimilar pair are used for computing the loss, for $i = 1, \ldots$ over training samples in the batch. However, usually the loss is aggregated for training samples in the batch, to make the loss value of the batch. Therefore the order doesn't matter; we can compute and aggregate the loss L_{sim} for training sample first. Then we can add the loss L_{dis} to compute the loss value of the batch.

The final trick is to use two different optimizers. For computing the loss L_{sim} for training samples, video frames are input the network to generate a caption. However, the loss L_{dis} is only used for learning the metric learning component, not for the encoder-decoder LSTMs. In other words, only v_1 connects the loss and the encoder and decoder components; v_2 is a given ground truth, and v_3, v_4 are only used for the metric learning component. Therefore we use different optimizers (updaters) for two losses. For training with the loss L_{sim}, one optimizer updates all networks weights. For training with the loss L_{dis}, another optimizer updates weigths in the metric learning component only.

4.5 Vocabulary

Our model uses word embedding to represent words. All the word inputs given to model is done so in the form of an embedding vector. So in essence each unique word in our vocabulary has been assigned to a 300 dimentional vector. Google wordtovec [10,11] provides these pre-trained embedding for words. It includes word vectors for a vocabulary of 3 million words and phrases that they trained on roughly 100 billion words from a Google News dataset. We select the most common words in the training captions and same from google wordtovec resulting in a vocabulary of 25231 words. Few words whose embeddings are not present in google wordtovec are initialized randomly from a normal distribution.

pre-trained vectors trained on part of Google News dataset [1]

5 Experiments

5.1 Dataset

The dataset used is the Microsoft Video Description corpus (MSVD) [16]. This is a collection of Youtube clips (1970 in total), average length of videos is about 6 seconds. Each video has descriptions from several annotators who describe the video in one sentence (40 captions per video on average). The data is split in the following way [1]; 1200 videos are used for training, and 100 for validation. The remaining 670 are used for testing. For a single video, we used up to $N = 80$ frames as input to the encoder LSTM.

5.2 Training procedure

The metric learning component uses the Siamese Manhattan loss, however we use a triplet loss for pre-training the the metric learning component. Here we can use the similar trick explained in the section of intra-batch training. A triplet loss takes three arguments; reference, positive, and negative samples. Given a batch of 50 samples, we have 50 pairs of generated and ground truth captions. For one of these pairs, the other 49 samples can be considered as negatives for the triplet loss. This is much more efficient than a naive training.

[1] https://code.google.com/archive/p/word2vec/

For the encoder and decoder components, we use the word-by-word loss without the metric learning component. Once the pre-training phase has been done, then we use the both losses for training in a stochastic manner. Specifically, given a batch, we randomly select if the metric leaning component is used (and then the Siamese Manhattan loss) in 70%, or not (the word-by-word loss is used) in 30%.

5.3 Results

The table 1 shows results of the baseline and proposed models. There are three different settings for the proposed models; O stands for the case when object information only (hence the attention and metric leaning component are not used), OA stands for the case when object information and attention is used but not the metric learning component, and OAM stands for the full model including the metric learning component.

The use of the object information clearly improve the performance against the baseline (original sequence-to-sequence) model. This is because main objects in each frame are explicitly used in the encoder.

Attention further improves the results (except CIDEr), which is expected as many results reported with better performance with attention.

Our full model, shown as OAM in Tab. 1, can further boost the performance by 1.1% in BLUE4. This is not as large as improvements with object information (by 10.7%) and attention (2.9%), but this results suggest that the proposed metric learning component can be used to improve results by adding to any models other than the sequence-to-sequence archtecture.

Table 1. Results of baseline and proposed models on the MSVD dataset. Symbols stand for; O – Objects, A – Attention, M – Meaning model.

Model	BLEU4	METEOR	CIDEr
Baseline (S2VT)	0.288	0.246	–
O	0.395	0.295	0.641
OA	0.424	0.312	0.641
OAM	0.435	0.316	0.649

Figure 3 shows some examples generated by the proposed model. In the first video (top row in the figure), the scene changes frequently because of the movie editing composing several cuts from different angles. Therfore the model without the metric learning component is confused and generated "dancing" instead of "riding".

In the last video (bottom row of the figure), the caption generated by the full model is considered as wrong because the ground truth caption mentions the animal as "animal" and it is obviously not a dog, while the generated caption says that it is a "dog". This is a limitation of the proposed model because the mistakes of the object detector ("dog" for "animal") diretly affects the encoder.

GT	A couple is on a bike.
O	a man is dancing.
OA	two people are dancing.
OAM	a man and a woman are riding a motorcycle.

GT	A man is lifting the car.
O	a man is lifting a car.
OA	a man is lifting the back of a truck.
OAM	a man is lifting a truck.

GT	An animal is eating.
O	a cat is licking a lollipop.
OA	a dog is eating.
OAM	a dog is eating.

Fig. 3. Examples of generated captions. Left images are frames of videos, and right texts are ground truth and generated captions. Symbols stand for; O – Objects, A – Attention, M – Meaning model.

6 Conclusions

We have proposed a model for video captioning guided by the similarity between captions. The proposed model has three components; the 2-layer LSTM encoder involving scene object information, the LSTM decoder with attention, and the metric learning for comparing two captions in a latent space. Experimental results show that the proposed model

Acknowledgement

This work was supported by International Linkage Degree Program (ILDP) in Hiroshima University (HU). This work was also supported by JSPS KAKENHI grant number JP16H06540.

References

1. Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R., Darrell, T., Saenko, K.: Sequence to sequence – video to text. In: 2015 IEEE International Conference on Computer Vision (ICCV). (Dec 2015) 4534–4542
2. Aafaq, N., Gilani, S.Z., Liu, W., Mian, A.: Video description: A survey of methods, datasets and evaluation metrics. CoRR **abs/1806.00186** (2018)
3. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. CoRR **abs/1804.02767** (2018)

4. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., eds.: Advances in Neural Information Processing Systems 30. Curran Associates, Inc. (2017) 5998–6008
5. Bai, S., An, S.: A survey on automatic image caption generation. Neurocomputing **311** (2018) 291 – 304
6. Hossain, M.Z., Sohel, F., Shiratuddin, M.F., Laga, H.: A comprehensive survey of deep learning for image captioning. ACM Comput. Surv. **51**(6) (February 2019) 118:1–118:36
7. Liu, X., Xu, Q., Wang, N.: A survey on deep neural network-based image captioning. The Visual Computer **35**(3) (Mar 2019) 445–470
8. Cornia, M., Baraldi, L., Cucchiara, R.: Show, control and tell: A framework for generating controllable and grounded captions. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (Jun 2019) 8307–8316
9. Li, Y., Duan, N., Zhou, B., Chu, X., Ouyang, W., Wang, X., Zhou, M.: Visual question generation as dual task of visual question answering. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (April 2018) 6116–6124
10. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2. NIPS'13, USA, Curran Associates Inc. (2013) 3111–3119
11. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings. (2013)
12. Luong, T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, Association for Computational Linguistics (September 2015) 1412–1421
13. Vinyals, O., Toshev, A., Bengio, S., Erhan, D.: Show and tell: A neural image caption generator. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (June 2015)
14. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing **45**(11) (Nov 1997) 2673–2681
15. Mueller, J., Thyagarajan, A.: Siamese recurrent architectures for learning sentence similarity. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. AAAI'16, AAAI Press (2016) 2786–2792
16. Chen, D.L., Dolan, W.B.: Collecting highly parallel data for paraphrase evaluation. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1. HLT '11, Stroudsburg, PA, USA, Association for Computational Linguistics (2011) 190–200