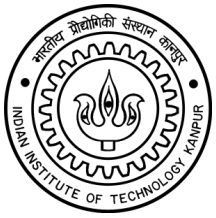
Image Captioning using Visual Attention



Indian Institute of Technology, Kanpur

Course Project-CS671A

Anadi Chaman(12616)
K.V.Sameer Raja(12332)

Mentor - Prof.Amitabha Mukerjee

Abstract

This aim of this project is to generate descriptive captions for images using neural language models. Deep learning has been used via a Convolutional Neural Network coupled with an LSTM based architecture. An image is passed on as an input to the CNN, which yields certain annotation vectors. Based on a human vision inspired notion of attention, a context vector is obtained as a function of these annotation vectors, which is then passed as an input to the LSTM. The LSTM network training is separately performed with randomly initialised word and phrase embeddings respectively. The output, thereby, obtained is passed to a softmax layer to generate captions. Our results have been evaluated using a metric named Meteor, and word embedding based implementation turns out to be better than its phrase embedding based counterpart.

Contents

1 Introduction

In this project, we intend to generate descriptive captions for images by utilizing neural language models. This task has been inspired by the work of Xu et.al [10] which utilizes a Convolutional Neural Network coupled with a Long Short Term Memory based architecture. A human vision inspired notion of attention has been incorporated in the model. We have sought to make some subtle changes in Kiros's work like using phrase embeddings instead of word embeddings for training LSTM. (We have observed that using phrase embeddings instead of their 'word' counterparts captures semantic and syntactic relations better in the generated captions.)
We have additionally compared the quality of captions generated for two cases: one where randomly initialised phrase embeddings are used in the LSTM training, and the second where embeddings trained on Google News Corpus have been used. Figure 1 illustrates an example of Image caption generation based on attention models.

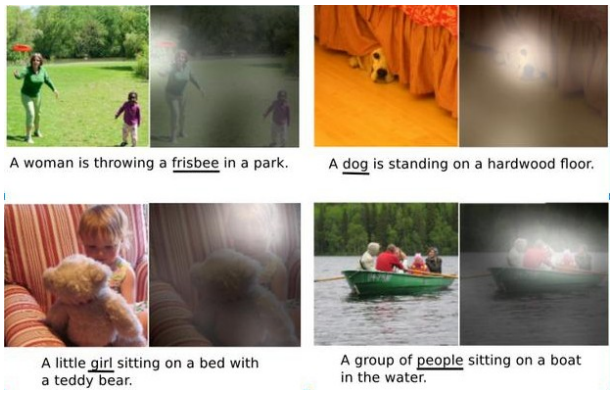


Figure 1: Image captions generated using Attention model
Source : Xu et al., 'Show, Attend and tell'

2 Motivation

Generating captions for images is a very intriguing task lying at the intersection of the areas of Computer vision and Natural Language Processing. This task is central to the problem of understanding a scene. A model used for generating image captions must be powerful enough that it can relate certain aspects of an image with the corresponding word in the caption generated.
Inspired by this idea, using attention models is an effective way of relating words with their relevant counterparts in the image. When a human observes an image scene, his/her attention is not static but dynamic which allows different components of the scene to come into forefront as and when required. Attention models work on this principle. Output of the CNN is used to dynamically shift attention to different parts of the image. As the attention shifts to these image parts, corresponding words are generated as the output of LSTM Architecture. Figure 2 illustrates the dynamic shift of attention and the corresponding generation of words.

3 Previous work

Traditionally, pre-defined templates have been used to generate captions for images. But this approach is very limited because it can not be used to generate lexically rich captions.

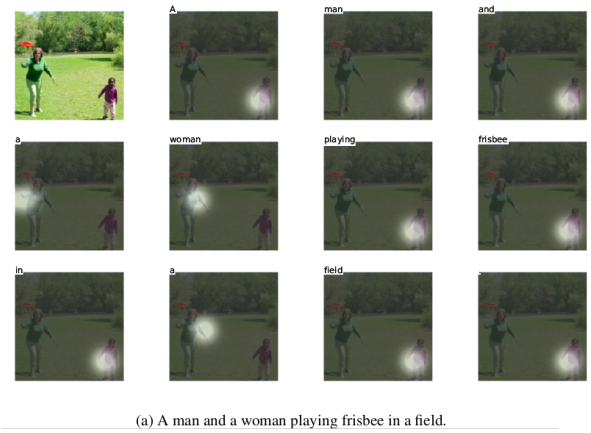


Figure 2: Visualising Attention model used for caption generation
Source : Xu et al., 'Show, Attend and tell'

The research in the problem of caption generation has seen a surge since the advancement in training neural networks and the availability of large classification datasets. Most of the related work has been based on training deep recurrent neural networks. The first paper that used neural networks for generating image captions was proposed by Kiros et al. [6], that used Multi-modal log bilinear model that was biased by the features obtained from input image.
Karpathy et.al [4] developed a model that generated text descriptions for images based on labels in the form of a set of sentences and images. They use multi-modal embeddings to align images and text based on a ranking model they proposed. Their model was evaluated on both full frame and region level experiments and it was found that their Multimodal Recurrent Neural Net architecture outperformed retrieval baselines.

4 Dataset

For training and validation purposes, we used Flickr8k dataset which contains 8000 images obtained from Flickr website. Corresponding to each image, five descriptive captions are available for training. Thus, in total, we trained our system on forty thousand captions.

5 Evaluation Metric

We use a metric named *METEOR* [2] for evaluating the performance of our system. *METEOR* scores a caption generated by our algorithm by aligning it with a reference caption. Precision and recall (for unigrams) are thereby calculated, and their harmonic mean multiplied with penalty is termed as *METEOR*. This alignment is achieved with three considerations: exact, stem and paraphrase matches. *METEOR* metric has certain free variables that have been tuned so as to produce a performance in coherence with various human judgement tasks like WMT ranking and NIST adequacy.
Figure 3 shows how *METEOR* is applied for scoring of a caption with respect to a reference caption.

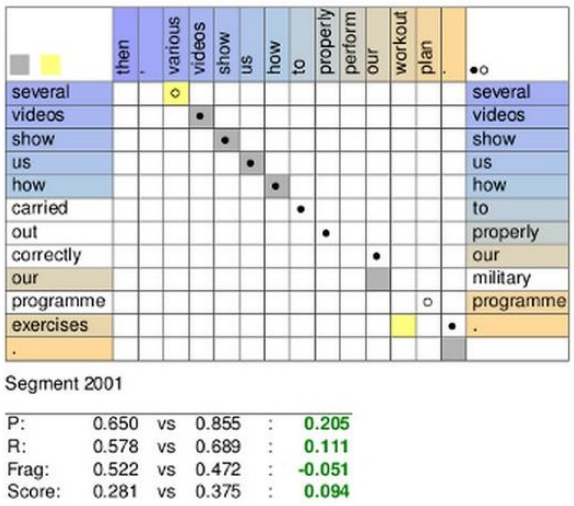


Figure 3: Example of *METEOR* applied for sentence scoring
Source : Xu et al., 'Show, Attend and tell'

6 Methodology

We have approached the problem of cation generation by two ways: using word embeddings and phrase embeddings. The implementation using word embeddings has been directly derived by the code released by Ryan Kiros [5]. On the other hand, phrase embeddings have been implemented by us. Both of these embeddings have been tested out, and their respective performances over Flickr8k dataset have been observed.
We can divide the overall approach for tackling the problem at hand in the following stages.

6.1 Incorporating Phrase Embeddings

Since, the implementation using word embeddings was directly available, the first task in front of us was to incorporate phrase embeddings in our algorithm. For this purpose, we have used SENNA software [1] to obtain phrases from the captions available to us, as a part of training data. For generating our captions, we are only using Noun, Verb and Prepositional Phrases, and neglecting others. This has been inspired by the statistics reported by Remi Lebret et al. [7]. Their work suggests that it is sufficient to use Noun, Verb and Prepositional Phrases to sufficiently capture the semantic meaning of a phrase.

Figure 4: Illustration of Chunking implemented using Senna Software

SENNA software performs chunking of a reference caption and yields the requisite phrases as output. The embeddings of these phrases are obtained by taking the sum of the embeddings of words within the phrase.
It can be noted from Fig 4, that the chunking output did not consider the adjectival phrase 'a pink'. It was however, simply embedded in a Noun phrase instead.

6.2 Annotation vector extraction

We use a pre-trained Convolutional Neural Network (CNN)[8] to extract feature vectors from input images. A CNN is a feed forward type of Artificial Neural Network which,unlike fully connected layers, has only a subset of nodes in previous layer connected to a node in the next layer.
Figure 5 illustrates the process of extraction of feature vectors from an image by a CNN. Given an input image of size 24×24, CNN generates 4 matrices by convolving the image with 4 different filters(one filter over entire image at a time). This yields 4 subimages or feature maps of size 20×20.These are then subsample to decrease the size of feature maps. These convolution and subsampling procedures are repeated at the subsequent stages. After certain stages these 2 dimensional feature maps are converted to 1 dimensional vector through a fully connected layer. This 1 dimensional vector can then be used for classification or other tasks. In our work, we will be using the feature maps (not the 1 dimensional hidden vector) called annotation vectors for generating context vectors.

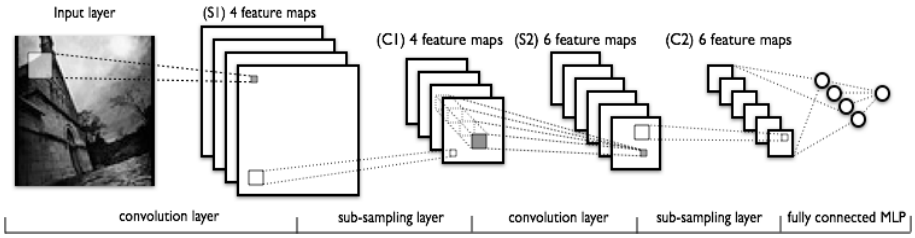


Figure 5: Feature map extraction using CNN

Source : http://deeplearning.net/tutorial/_images/mylenet.png

6.3 Context Vector generation

Based on the annotation vectors extracted, a context vector is generated using visual attention models. The attention models are inspired by the way humans attend to their surroundings. When a human looks at his/her surroundings, the information is not processed in a static form. Humans keep on dynamically focusing on different parts of the image scene, with different aspects of the scene appearing in the forefront of focus as and when required. The attention models proposed by Xu et al. also work in a similar fashion. An image is taken as input, and a context vector is defined that focuses on a particular aspect of the image. This focus keeps on shifting dynamically to different aspects of the image, and corresponding captions are generated. There are two variants of attention models that have been proposed by Xu et.al [10]: Hard attention and Soft attention. Hard attention is a

stochastic attention mechanism that is trained by maximising a variational lower bound of log likelihood function, whereas 'soft' attention is a deterministic mode of attention that is trained by standard back propagation.
According to Karpathy [3], soft attention model is convenient to use because it keeps the entire model fully differentiable, but it is not very efficient. Based on this idea, we have only considered using Hard Attention models for generating context vector.

Hard Attention: Consider an image passed as an input to our CNN. The output gives us annotation vectors of size 512 corresponding to 196 different portions of the image. Let $\hat{z_t}$ denote the context vector at time instant t. Let s_t denote a location variable when the t^{th} word is being generated. s_t has a one hot representation, where $s_{t,i}$ is 1 when the i^{th} location is being attended. These location variables are considered intermediate location variables, and a mutlinoulli distribution is assigned to them, as:

$$p(s_{t,i} = 1 | s_{j<t}, a) = \alpha_{t,i} \tag{1}$$

$\hat{z_t}$ is thereby defined as,

$$\hat{z_t} = \sum_i s_{t,i} a_i \tag{2}$$

It can be seen that this definition in Eq. (2) defines $\hat{z_t}$ as a random variable. Now, consider the log likelihood function $log(p(y|a))$ as the likelihood function of generating a sequence of words y, given a feature set a. This likelihood function needs to be maximised with respect to the model weights W. This maximisation is achieved indirectly by maximising a different objective function L_s, which is a variational lower bound of this function.

$$\begin{aligned} \mathrm{L}_s &= \textstyle\sum_s p(s|a) log(p(y|s,a) \\ &\leq log \textstyle\sum_s p(s|a) p(y|s,a) \\ &= log(p(y|a) \end{aligned} \tag{3}$$

Eq. (3) shows that L_s is a lower bound of the log likelihood function, and Eq. (4) suggests a Monte Carlo based sampling method that is used to approximate the gradient of L_s with respect to model parameters. This is achieved by sampling s_t by using a multinoulli distribution as shown below.

$$\widetilde{s_t} = Multinoulli_L(\alpha_i) \tag{4}$$

$$\frac{\partial L_s}{\partial W} \approx \frac{1}{N} \times \sum_{n=1}^{N} \Big[\frac{\partial log(p(y|\widetilde{s^n}, a)}{\partial W} + log(p(y|\widetilde{s^n}, a)) \times \frac{log(p(\widetilde{s^n}|a))}{\partial W} \Big] \tag{5}$$

The variance in the Monte-carlo estimator is reduced by using a Moving average baseline described in [9]. Finally, the model is learnt using the rule:

$$\frac{\partial L_s}{\partial W} \approx \frac{1}{N} \times \sum_{n=1}^{N} \Big[\frac{\partial log(p(y|\widetilde{s^n}, a)}{\partial W} + \lambda_r (log(p(y|\widetilde{s^n}, a)) - b) \frac{\partial log(p(\widetilde{s^n}|a))}{\partial W} + \lambda_e \frac{\partial H[\widetilde{s^n}]}{\partial W} \Big] \tag{6}$$

In Eq. 7, λ_r and λ_e are two hyper parameters obtained by performing cross validation.

6.4 LSTM Network

A Long Short Term Memory Network is used to generate captions at t^{th} time instant given the captions generated previously and the context vector at time t. We have followed the LSTM implementation as used by Xu et al. Figure 6 illustrates an LSTM cell used by Xu et al. They have used an affine transformation $T_{s,t} : R_s \rightarrow R_t$ whose parameters are learned during the training of LSTM network.

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ g_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} T_{D+m+n,n} \begin{bmatrix} Ey_{t-1} \\ h_{t-1} \\ \hat{z}_t \end{bmatrix} \tag{7}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{8}$$

$$h_t = o_t \odot tanh(c_t) \tag{9}$$

In Equation 7, 8 and 9, i_t, f_t, c_t, o_t, h_t denote input, forget, memory, output and hidden state variables. The variable $\hat{z}_t \in \mathbb{R}^D$ is the context vector at time instant t, $E \in \mathbb{R}^{m \times K}$ is an embedding matrix. In our work, E has been used to include both word and phrase embeddings. σ is sigmoidal function and $\odot$ denotes element wise multiplication. The initial values of c and h are determined by taking an average of annotation vectors that are obtained from two Multi-Layer Perceptrons.

$$c_0 = f_{init,c}\left(\frac{1}{L}\sum_{i}^{L} a_i\right)$$

$$h_0 = f_{init,h}\left(\frac{1}{L}\sum_{i}^{L} a_i\right)$$

Here, L is the number of annotation vectors yielded by CNN.

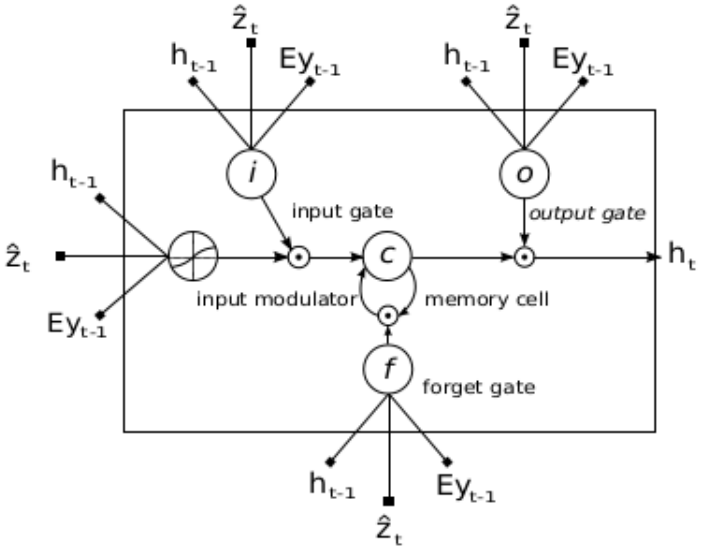


Figure 6: An LSTM cell

6.5 Training Procedure

- For annotation vectors, we used a pre-trained model of CNN namely Oxford VGGnet trained on Imagenet dataset.
- We are using a LSTM network with 1 hidden layer having 1000 cells. As proposed by Xu et.al, we use mini batches containing captions of same length while training. RMSprop with an adaptive learning rate has been used for weight updation. The base code we are using is that of Xu.et.al [5] with modifications made for using phrases.
- Due to lack of time, we stopped the training after 238 epochs. To compare the results, we trained the model with words as input (same as Xu.et.al work [5]) and halted it after 225 epochs to get a reference for comparing the results.
- Due to large vocabulary size of phrases, we computed the most frequently occurring ones and replaced the rest of the word with a special word (UNK). This reduced our vocabulary to 10000 phrases. The reasoning behind deleting less frequent phrases, is that those phrases are often the modified forms of other phrases. So we made an assumption that the architecture would learn to replace these UNK words with reasonable phrases.
  For ex : frequent phrase : A dog
  less frequent phrase : A white dog with black spots
- For experimenting using pre initialised embeddings of phrases, we use a pre-trained word2vec model trained of google news corpus having 3 billion words. The vector embedding of the phrase is derived by summing the vector embeddings for each word in the phrase.

6.6 Beam search for choosing captions

For choosing captions in addition to the conventional way, we implemented Beam search algorithm as well at the output of LSTM network. Each candidate caption has a certain probability of occurence. Instead of choosing just the one with the highest probability, Beam search keeps a list of K best candidate choices at the output.

7 Results

After training the model for both the word and phrase embedding cases, we compared the captions generated by both with reference captions that we had in our dataset. As descried in Section 6, we used *METEOR* evaluation metric to score the quality of captions generated. Table 1 illustrates the *METEOR* score we obtained with our implementations.

Table 1: *METEOR* score obtained on evaluation using word and phrase embeddings

Input	vocabulary	METEOR
phrases	10000	0.062
phrases(pre initialised embeddings)	10000	0.0595
phrases	36220	0.041
words	9630	0.067
words (using beam search)	9630	0.089

- We used a vocabulary size of 9630 for words, and used two different ways of finding captions using this vocabulary-one where we did not use beam search, and the other where we did. Table 1 indicates that using beam search at the output of LSTM yielded better captions, since the corresponding *METEOR* score is the highest for it.
- For phrase embeddings, we used two different vocabulary sizes: 10,000 and 36, 220. Here, captions generated using the smaller vocabulary size (10,000) were found to be better, having a *METEOR* score of 0.062 in comparison with a score of 0.041 as obtained from the larger vocabulary.

Figure 7, 8 and 9 shows the value of the error function on the test dataset, plotted against the number of epochs for word, phrases with 10000 vocabulary and phrases with 36220 vocabulary as inputs repectively. For the implementation using word embeddings, the error starts from 64 and begins to fall until it reaches 45. For the implementation based on phrase embeddings, the error function does not fall by any reasonable extent when the vocabulary used has a size of 36220. It stays almost equal to 48. On the other hand, with a phrase vocabulary of 10000, the error falls from 41 to approximately 37. Figure **??** shows an example of captions generated for an image.

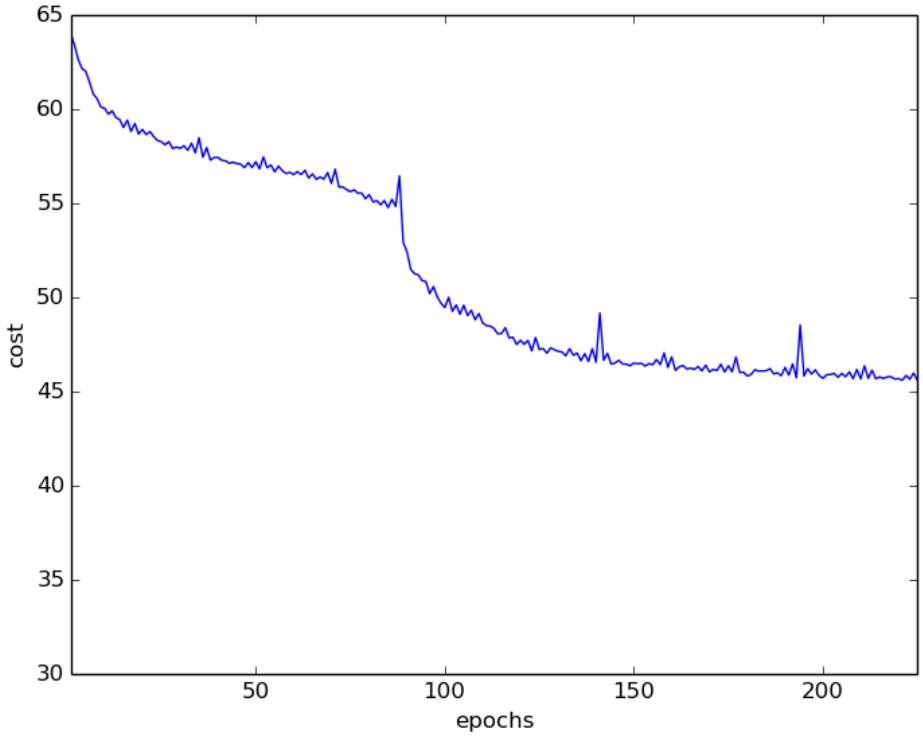


Figure 7: Cost change with epochs for word inputs

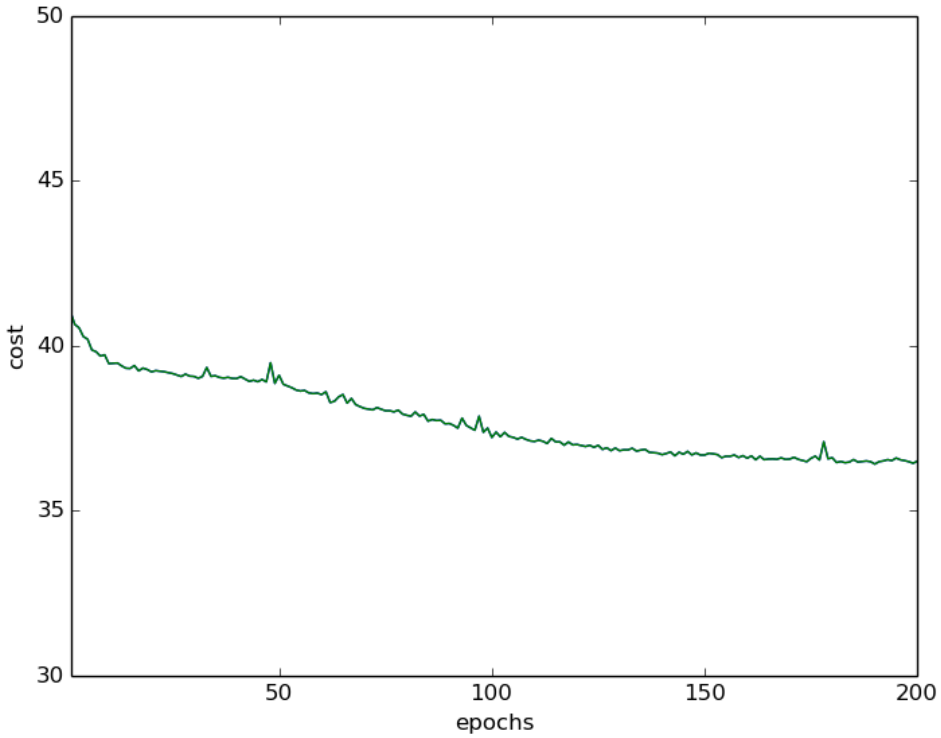


Figure 8: Cost change with epochs for phrase inputs when vocabulary size is 10,000

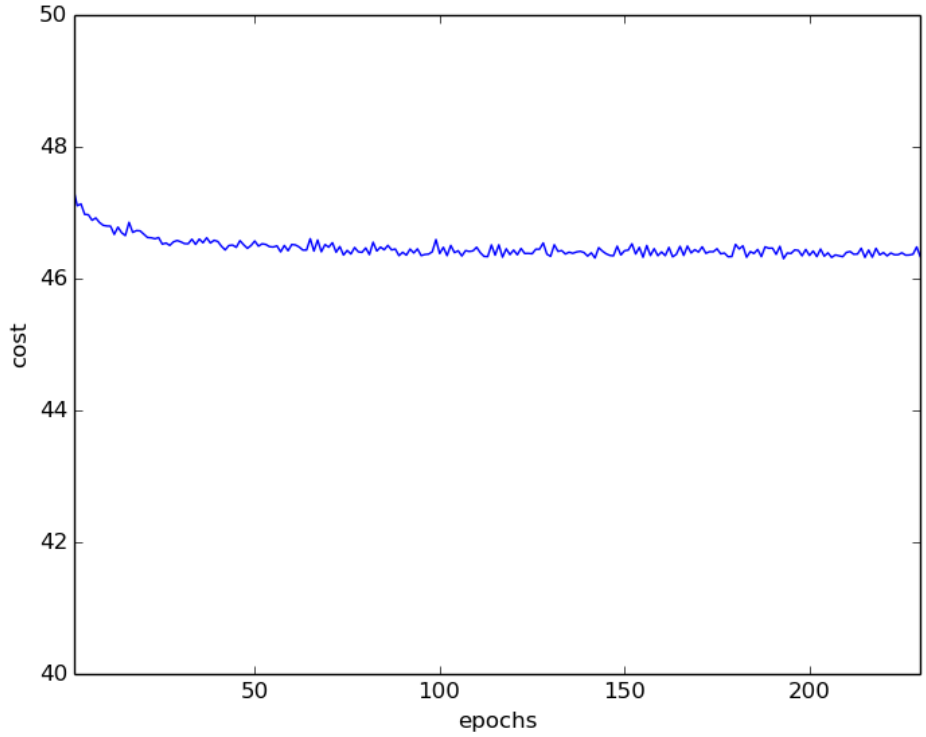


Figure 9: Cost change with epochs for phrase inputs when vocabulary size is 36,220

8 Inferences

From the results that we obtained, the following inferences can be made.

- In their implementation using hard attention, Xu et al., obtained METEOR score of 0.203 on Flickr8k dataset, whereas our result turned out to be 0.089. Even though the overall architecture that we used was derived from theirs itself, this reduced *METEOR* score can be explained by the difference in the number of epochs that we and they used. Our model was trained on 225 epochs, whereas theirs was trained on five thousand.

- The two tests on different phrase vocabulary sizes, show a surprising difference. This difference can most likely be attributed to the problem of over-fitting. Given that we had a training set of 8000 images with 5 captions per image, the total number of captions being used for training is 40000. Using only 40000 captions for training a model with a vocabulary size of 36,220 (caption to vocabulary ratio being 1.1) is very likely to lead to over-fitting, and thus poor quality of results. On the contrary, when a vocabulary size of 10,000 is used (caption-vocabulary ratio being 4), overfitting is not that likely to occur.

- Initialising the phrase embedding matrix by pre-trained word vectors doesnot bring much change in accuracy.

- The results on word embeddings indicate that applying Beam search on LSTM output make the captions more informative. However, this claim is still debatable as the number of epochs over which our model was trained was not reasonably large.

9 Conclusions

- **Phrase vocabulary reduction**: The marginal decrease in accuracy of phrases over words is due to large number of UNK symbols in training data which is resulting in UNK symbols in generated captions and hence low score. So instead of deleting low frequency phrases, we can use some other phrase vocabulary reduction techniques to account for UNK symbols. This may probably increase the accuracy of using phrases as input over words.
- **Epochs** : As only 225 epochs are completed, it is difficult to comment on the absolute accuracies. So we used the original model (using words) by Xu.et.al, which we ran for 225 epochs and then compared the accuracies between phrases and words as inputs. If the rate of decrease in error continues in the same manner, then we can expect improved accuracy of generated captions using phrases as inputs.
- Beam search has improved accuracy for words whereas for phrases the generated captions are making no sense. The length of the generated captions is not more than 4 in most of the cases. Again as we are using less number of epochs, if is difficult to comment as beam search uses set of learned weights (probabilites) for next word/phrase generation.

Caption generation results

Figure 10: **Actual** : A tan dog jumps into water
Using word as input : A brown dog is walking in the water
Using phrase as input : a dog walking a shallow river looking into in the water gray shorts the horizon with in down water

Figure 11: **Actual**:A brown and white dog running in a field covered in yellow flowers
using word as input: A black dog running Frisbee in galloping
using phrase as input :a white and brown dog in his face a field

Figure 12: **Actual**: a black and white dog is running on the beach
using word as input:Man running along the beach , running and a beach
using phrase as input:a black and white dog running UNK playing on a coffee shop a skate park

Figure 13: **Actual**:A young boy leaps off a swing
using word as input: A girl with a measures is standing in a parking field
using phrase as input :a man holding in is jumping a swing performs

10 Acknowledgement

We thank Prof. Amitabha Mukerjee, Dept. of Computer Science and Engineering for his valuable support throughout the project guiding us from time to time and looking into the project when it was needed.

References

[1] COLLOBERT, R., WESTON, J., BOTTOU, L., KARLEN, M., KAVUKCUOGLU, K., AND KUKSA, P. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research 12* (2011), 2493–2537.

[2] DENKOWSKI, M., AND LAVIE, A. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation* (2014).

[3] KARPATHY. The Unreasonable Effectiveness of Recurrent Neural Networks. `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`.

[4] KARPATHY, A., AND FEI-FEI, L. Deep visual-semantic alignments for generating image descriptions. *arXiv preprint arXiv:1412.2306* (2014).

[5] KELVINXU. arctic-captions. `https://github.com/kelvinxu/arctic-captions`.

[6] KIROS, R., SALAKHUTDINOV, R., AND ZEMEL, R. Multimodal neural language models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)* (2014), T. Jebara and E. P. Xing, Eds., JMLR Workshop and Conference Proceedings, pp. 595–603.

[7] LEBRET, R., PINHEIRO, P. O., AND COLLOBERT, R. Phrase-based image captioning. *arXiv preprint arXiv:1502.03671* (2015).

[8] SIMONYAN, K., AND ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *CoRR abs/1409.1556* (2014).

[9] WEAVER, L., AND TAO, N. The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence* (2001), Morgan Kaufmann Publishers Inc., pp. 538–545.

[10] XU, K., BA, J., KIROS, R., CHO, K., COURVILLE, A. C., SALAKHUTDINOV, R., ZEMEL, R. S., AND BENGIO, Y. Show, attend and tell: Neural image caption generation with visual attention. *CoRR abs/1502.03044* (2015).