Thesis Progress Overview February

Wout Vekemans & Thijs Dieltjens

February 27, 2016

1 Karpathy's code

Karpathy hosts two implementations on his GitHub page. The first one is an implementation of his own RNN and the second one is an implementation based on the paper on LSTM by Vinyals. We cloned both implementations and trained them on the Flickr30k dataset. The results of this training process will be used as a baseline for our experiments.

Karpathy uses the output of the last layer of a CNN when fed an image, as the vector representation for this image. This convolutional neural network is implemented in Caffe. The RNN implementation (described in [Karpathy and Fei-Fei, 2015]) feeds an image representation to an RNN, together with a representation of the reference sentence. The network is then trained to predict a sequence of words, based on a given image representation. The training is done using backpropagation. To generate a new sentence, beam search is used.

The LSTM implementation based on [Vinyals et al., 2015] adds memory blocks to an RNN. These blocks use gates to control when a value should be forgotten. This model is then trained to predict a word based on the image representation and all previous words. The image is only fed to the network at the beginning, since feeding it at each timestep yields inferior results.

Both of the Karpathy implementations use perplexity calculated on the validation set to evaluate the results after each epoch. To keep the network from exploding, the perplexity for each batch is also calculated after it is processed, and the training is stopped when this becomes to high too quickly.

2 Ideas considered

2.1 FSMN

One of the first additions we tried to make to the Karpathy code was the introduction of a feedforward sequential memory neural networks (FSMN). The reason we tried this is the claim of the original paper [Zhang et al., 2015] that this network could be trained more quickly than an LSTM. The paper however was not clear on the entire implementation of the network, so we kept a few parameters fixed just to check the speed. We fed the image vector in

the same way as Karpathy's CNN implementation. Running this preliminary code did not bring the speed-up we were hoping for, so we stopped the further implementation of this network.

2.2 Flickr30K Entities

The Flickr30k Entities paper [Plummer et al., 2015] provides a dataset containing region-to-phrase correspondences for the original Flickr30k dataset. These dense annotations contain a lot of useful information. We tried to incorporate this information in our models doing the following. We applied a filter on the image regions, leaving out all those that are smaller than 64 by 64 pixels, since they would be stretched to 256 by 256 when computing the image representation. This stretching would lead to unclear, blurry images. This set of filtered images is fed through a CNN (VGGNet) using Caffe. The corresponding phrases are represented using tf-idf weighted vectors. Using these two sets (CNN features and tf-idf phrase vectors) we tried to compute the stacked CCA (as explained in our presentation). We tried to use stacked CCA to create an "improved" image representation, that incorporates the extra information that is provided by the entities. This improved representation can be used as a guide to the gLSTM.

It turned out to take extremely long to compute this, so we stopped following this route.

2.3 CCA

To provide the system with additional semantic information, we are currently looking into using a simple application of CCA. The used mapping is based on the image representations provided by the CNN, and tf-idf vectors of the corresponding descriptions. Each image is used five times, once for each of the captions. Based on these two sets, a CCA model is trained. At training time, the CCA projection of the image into the multimodal space is used as a guide for the gLSTM. Currently we are experiencing difficulties calculating the CCA projection, after looking into two different Python implementations (scikit-learn and PyRCCA), both yield errors, or freezing executions.

2.4 LDA

One of the things we worked on the most is using LDA as additional semantic information.

We first trained an LDA model on the training set. For each image, we concatenated the five sentences, removed stopwords and stemmed each word. As the amount of topics we used 80 and 120, these amounts result from manual optimization. When using 50 topics, the top 10 words for each topics were not related enough to have a distinct topic. Using more than 120 topics lead to a lot of very similar topics, which is not very useful. Based on the topic-word distributions resulting from this training, we calculated the topic distribution for the test and validation set.

To see if this was an approach that has useful results, we manually validated it. The first thing we checked was the word distribution for each topic: using the top 10 words for each topic, we tried to label each topic. This resulted in quite natural topic descriptions, with a few (5-10) topics that could not be labeled directly. To validate the network, we checked the prediction for a subset of the test images. We checked which topics scored the highest (top 5) and compared this result with what is on the actual image. We checked around 50 images, and in approximately 40 of the cases, at least 4 of the topics were relevant to the image.

Using the topic distributions of the training images, we trained a multi-input multi-output perceptron to predict the topic distribution for a given image. To validate this system we used the negative log probability on the validation set. After training this network, the predictions for the test and validation set are stored, since they will be used for evaluation purposes. Doing this, we have a topic distribution for each image: for the training set this is a direct output of the LDA model, while the test and validation set distributions are calculated by the perceptron.

3 Implementation

3.1 RNN with LDA

To improve the performance of Karpathy's RNN implementation, we added the additional information that the LDA topic distribution provides to the training procedure. This is done the same way as the image representation is used.

The image and topic distribution are added at the first timestep. We have the option included to keep adding it every timestep, but have not yet experimented with that. Karpathy however does report that repeatedly feeding the image to the network yields worse results. The paper of Jia reports that using the image as a guide performs worse, whereas smaller guides (e.g. CCA projection of the image) perform better. With this in mind we will certainly implement a RNN network that gets the LDA distribution as an input at each step.

3.2 gLSTM

A recently published paper [Jia et al., 2015] proposes two modifications to the Karpathy implementation of the LSTM. A first modification is the addition of a semantic guide to the LSTM network. They coined the term gLSTM for this type of network. A guide is a vector that contains semantic information about the image and is fed into the LSTM network at each time step. Most of the guides they propose improve the scores of the system. The best results were obtained by using the vector produced by doing a CCA mapping of the image. Secondly they noticed the generated sentences favor short sentences. To solve this problem they propose four different length normalization methods that can be put into the beam search.

In our opinion the addition of semantic information during the generation seems a logical step to avoid for example semantic drift. Therefore we also implemented the gLSTM. We then tried to use CCA in the same way as the paper, to see if we can get similar results. At this time however, we have not yet been able to train the CCA so this implementation can not yet be used. A second piece of semantic information we then tried to use is the LDA topic distribution that has been explained in the previous sections.

4 Evaluation

4.1 Evaluation Metrics

To evaluate a trained model, the model first gets each test image as an input and tries to generate a sentence describing it. The evaluation is done by comparing the generated sentence to the five reference sentences. This can be done by using several metrics that were originally designed for machine translation systems. The goal of each of these metrics is to judge the quality of the translated sentences based on both their content and fluency.

A first metric that is used throughout the literature is the BLEU score.[Papineni et al., 2001] This score mainly looks at the n-grams precision. This precision can be calculated for each number of ngrams. The B@n score is computed as a geometric mean of these precisions. The original paper additionally proposes to use a brevity penalty which gives shorter sentences a lower score.

A second metric that can be used and has been shown to correlate more with human behaviour is the METEOR score as introduced in [Denkowski and Lavie, 2007]. METEOR tries to align the generated sentence with the reference sentences. It does this by matching both words and phrases and by considering synonyms and performing stemming. It accounts for both precision, recall and the importance of grammaticality.

As a final way of evaluation we look at the length of the sentences and the amount of different words with their number of occurences. We believe that both of these metrics can be used to examine the quality of the generated sentences. By looking at the number of sentences of each length, it is for example possible to examine the number of sentences of length two. These sentences can never be considered as a good description for the image. By looking at the number of different words, the expressiveness of the model can be checked and compared to the references. By looking at the number of occurences of each word, they can be ranked. This ranking offers useful insights into for example which words are used more (or less) than a human generated sentence.

4.2 Problems with evaluation

This section discusses some general problems and difficulties we faced with the previously described metrics.

Firstly the BLEU score has been reported to favor short sentences (especially without the brevity penalty). Several papers also mention that there is only low correlation with human judgements. Secondly in the literature some papers use the aforementioned brevity penalty, while others do not. This is not always reported, which makes comparing results from different papers quite difficult. The brevity penalty only looks at the length of the shortest reference while ignoring the longer ones, so it is perhaps not the best way of penalizing short sentences.

Then we also noticed that given the same input, different implementations of BLEU can give different answers. At the moment it is not really clear what causes this except for a bug we discovered in the library we used. Therefore we decided to stick with the implementation that Karpathy provides for the final evaluation of our models.

Finally we are currently facing a problem with the METEOR scores. The code accompanying the paper is freely available. This way it is possible for us to compute a METEOR score. Currently however these scores seem too low when compared to the literature. We believe this is caused by different settings of this computation. In the following weeks we will try to figure out which settings other papers use to calculate this METEOR score and try to fix this mismatch.

5 Results

Table 1 gives an overview of our results so far compared to the results as shown in the literature. Note that our results for LSTM should roughly be the same as that of NIC. NIC also uses ensemble methods however, which we do not. The same should be found for Karpathy and our RNN. We notice however that there is a mismatch here. We believe this may be caused because Karpathy uses the brevity penalty for computing the Bleu score. When using a common brevity penalty of around 0.9, the results for the Karpathy paper become close to ours. In the paper of Xu for example they explicitly state however they set the brevity penalty to one. Throughout the literature both options are used. In our results we also chose to put the brevity penalty to one.

The results we present here are only preliminary since each of these models is still training. Both RNN and LSTM have both had more than 6 days of training. The RNN with LDA has had roughly 4 days of training. The gLSTM model with LDA has only had a remarkable 1 day of training.

Even with preliminary results, the RNN+LDA model reaches results that are on par with the current state-of-the-art.

We have not had the time to check the word occurences and sentence lenghts of each of the models but we already see some common characteristics. Each of of the models uses only 200-300 different words which is low compared to the training vocabulary of around 5000 words. In normal sentences we expect to see mostly stopwords among the most occurring words. With our generated sentences this is not entirely the case. As an example, in most of the models

	B1	B2	B3	B4	METEOR
Karpathy (RNN)	57.3	36.9	24	15.7	
Google NIC (LSTM)	66.3	42.3	27.7	18.3	
Jia (gLSTM + gaussian norm.)	64.4	**44.6**	**30.5**	**20.6**	17.91
Jia (gLSTM + polynom. norm.)	59.8	41.3	29.3	19.2	**18.58**
Xu (attention)	**66.9**	43.9	29.6	19.9	18.46
RNN (ours)	64.9	43.1	28.1	17.8	
LSTM (ours)	62.1	41.4	27.1	17.6	
RNN+LDA 120	66	**44.6**	29.2	18.7	
gLSTM (LDA 80)	61.8	40.7	26.6	17.1	

Table 1: Comparison of the results on the Flickr30k set for our models and those in the literature (above). Bn is the B@n Bleu score. *LDA x* means a model with x topics is used.

we tested so far, the word 'man' is in the top five. The opposite word 'woman' however occurs about eight times less. It is therefore possible that the models will have a bias towards 'man' when faced with the image of a person.

6 Future

6.1 Attention Based

The next weeks we will look into attention based models, and try to find a way to incorporate this in one of the models we already have. The literature shows promising results, as seen in table 1 where the scores for Xu are comparable to those of Jia.

6.2 Beam Search Normalization

Besides the introduction of gLSTMs, Jia also introduces length normalization techniques to the beam search that improve the results. A possible extension of our current code can be the addition of one or more of the most performant normalization strategies.

6.3 Further Optimizations

We will first continue training our current models, until there is no more improvement on the perplexity on the validation set.

Secondly we will also look at the effect of the number of topics in the LDA by training the gLSTM using 120 topics and the RNN with 80 topics.

At this time the rnn gets fed the lda values only at the beginning of a sentence. It might be possible that feeding the lda value at each step improves the results. This is a small change to our code that is worth looking into.

Finally we will try to find a solution for the problems with calculating of the CCA.

References

[Denkowski and Lavie, 2007] Denkowski, M. and Lavie, A. (2007). Meteor Universal : Language Specific Translation Evaluation for Any Target Language.

[Jia et al., 2015] Jia, X., Gavves, E., Fernando, B., and Tuytelaars, T. (2015). Guiding Long-Short Term Memory for Image Caption Generation.

[Karpathy and Fei-Fei, 2015] Karpathy, a. and Fei-Fei, L. (2015). Deep Visual-Semantic Alignments for Generating Image Des. *Cvpr2015*.

[Papineni et al., 2001] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2001). BLEU: a Method for Automatic Evaluation of Machine Translation. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, (July):311–318.

[Plummer et al., 2015] Plummer, B. a., Wang, L., Cervantes, C. M., Caicedo, J. C., Hockenmaier, J., and Lazebnik, S. (2015). Flickr30k Entities: Collecting Region-to-Phrase Correspondences for Richer Image-to-Sentence Models.

[Vinyals et al., 2015] Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and Tell: A Neural Image Caption Generator.

[Zhang et al., 2015] Zhang, S., Jiang, H., Wei, S., and Dai, L. (2015). Feedforward Sequential Memory Neural Networks without Recurrent Feedback.