Decoding Strategies for Neural Referring Expression Generation

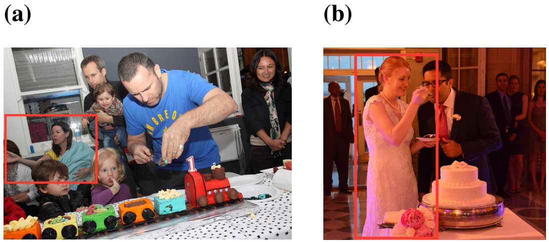
Sina Zarrieß and **David Schlangen**
Dialogue Systems Group // CITEC // Faculty of Linguistics and Literary Studies
Bielefeld University, Germany
{sina.zarriess,david.schlangen}@uni-bielefeld.de

Abstract

RNN-based sequence generation is now widely used in NLP and NLG (natural language generation). Most work focusses on how to train RNNs, even though also decoding is not necessarily straightforward: previous work on neural MT found seq2seq models to radically prefer short candidates, and has proposed a number of beam search heuristics to deal with this. In this work, we assess decoding strategies for referring expression generation with neural models. Here, expression length is crucial: output should neither contain too much or too little information, in order to be pragmatically adequate. We find that most beam search heuristics developed for MT do not generalize well to referring expression generation (REG), and do not generally outperform greedy decoding. We observe that beam search heuristics for termination seem to override the model's knowledge of what a good stopping point is. Therefore, we also explore a recent approach called trainable decoding, which uses a small network to modify the RNN's hidden state for better decoding results. We find this approach to consistently outperform greedy decoding for REG.

1 Introduction

Recently, many NLP problems that involve some form of natural language generation have been modeled with encoder-decoder architectures based on recurrent neural networks, e.g. in machine translation (Bahdanau et al., 2014), summarization (Ranzato et al., 2016), conversation modeling (Vinyals and Le, 2015), image captioning (Xu et al., 2015), etc. A simple and efficient way to apply the decoder model during testing is to generate the most likely word at each time step, until an end symbol has been generated or the maximal number of time steps has been reached. However, this greedy search method does not generally produce optimal generation output, and has been shown to produce repetitive or invariable sentences in certain tasks, e.g. (Li et al., 2016). Thus, a common practice (particularly in MT) is to use beam search where a fixed number of hypotheses are considered (and expanded) at each time step. Unfortunately, beam search is a heuristic algorithm that can be defined and parametrized in different ways, and brings with it further modeling decisions that need to be explored. A particularly tricky issue in neural MT, for instance, is to define a good stopping criterion for search, as neural models tend to radically prefer short hypotheses (Graves, 2012). To the best of our knowledge, it has not yet been systematically investigated whether beam search heuristics developed for MT carry over to other generation tasks, in particular, to tasks in the area of language and vision.



refexp: woman in light blue background left

refexp: lady

Figure 1: RefCOCO (Yu et al., 2016) examples for referring expressions to targets (red box) in images

In this paper, we investigate decoding strategies for neural referring expression generation (REG) applied to objects in real-world images. This task is closely related to image captioning in the sense

Proceedings of The 11th International Natural Language Generation Conference, pages 503–512,
Tilburg, The Netherlands, November 5-8, 2018. ©2018 Association for Computational Linguistics

that a visual entity has to be described by a verbal, semantically adequate expression. But beyond semantic adequacy, REG also requires pragmatic reasoning: referring expressions typically do not only depend on the object they refer to, but also on the visual context of that object, as human speakers tend to tailor their utterances such that a listener can easily understand them in the current context. For instance, a short RE that simply names the object (e.g. *lady* in Figure 1(b)) is unlikely to be produced than in a scene that contains more objects of the same category, see Figure 1(a). Thus, previous work on neural REG has investigated techniques of incorporating contextual knowledge into the model during training, looking at different visual representations and optimization techniques (Mao et al., 2015; Yu et al., 2016). Somewhat surprisingly, however, relatively complex architectural set-ups are needed to improve over simple, context-agnostic baselines that generate descriptions for the objects as captioning models do for entire images (Yu et al., 2017).

In this paper, we take a closer look at the decoding step in neural REG. As referring expressions tend to be much shorter than full sentences in MT, for instance, it is not guaranteed that heuristics developed for beam search in MT are equally successful. More importantly even, the problem of determining the appropriate length of a referring expression, i.e. terminating beam search, is conceptually different than determining the length of a good translation: a translation is complete when it covers the *meaning* of the words in the source sentence, and indeed, the length of the source is used as criterion in beam search for MT (see Section 2.3). A referring expression, on the other hand, is complete when it describes the visual target in a *pragmatically* adequate way, i.e. when it neither provides too little nor too much information.

We explore a range of different variants of beam search that have been proposed for MT and, interestingly, find that most of them *decrease* performance as compared to simple greedy decoding in REG. Whereas greedy decoding leads to referring expressions that, on average, have an adequate length, beam search produces REs that are markedly shorter and various heuristics can only partially remedy for this problem. Therefore, we look at *trainable decoding*, a method proposed by Gu et al. (2017), that offers a principled solution for obtaining a decoder that maximizes a given objective (e.g. BLEU scores). This method has been shown to outperform greedy decoding and to be computationally more efficient than beam search in MT (Gu et al., 2017; Chen et al., 2018). We find that it qualitatively outperforms both greedy and beam search in the case of neural REG.

2 Related Work

2.1 Symbolic formalizations of REG

Traditionally, research on NLG has conceived of REG as a multi-stage process that involves the tasks of lexicalization, content selection and surface realization (Reiter and Dale, 2000; Krahmer and Van Deemter, 2012). But foundational work in REG has mostly focused on algorithms for attribute selection (Dale and Reiter, 1995). In this paradigm, the task is to generate a distinguishing referring expression for an object in a visual scene, which is defined as a target object, a set of distractor objects and a set of symbolic attributes (e.g. type, position, size, color, ...), see Krahmer and Van Deemter (2012). In this setting, an attribute is said to *rule out* a distractor object, if the target and distractor have different values. Dale and Reiter (1995)'s well-known *Incremental Algorithm* (IA) iterates over the attribute set in a pre-defined order, selects an attribute if it rules out objects from the set of distractors and terminates when the set is empty. In the context of our work, this algorithm can be seen as a decoding procedure over a symbolically specified input for REG that heuristically defines the stopping criterion, i.e. when to terminate the expansion of the RE. A lot of subsequent work has looked at refining and extending this algorithm, testing it on human-generated expressions (Krahmer et al., 2003; Mitchell et al., 2010; van Deemter et al., 2012; Clarke et al., 2013).

2.2 Neural REG from real-world images

More recently, research on REG has started to investigate set-ups based on real-world images (Kazemzadeh et al., 2014; Gkatzia et al., 2015; Mao et al., 2015; Zarrieß and Schlangen, 2016), representing scenes with many different types of real-world objects. Here, the input to the REG system is defined as a low-level visual representation such that various aspects of the task have to be addressed, including lexicalization and content selection. Inspired by research on image captioning, Mao et al. (2015) proposed the first neural end-to-end model for REG that uses a CNN to represent the image, followed by an LSTM to generate the referring expression. Yu et al. (2016, 2017) investigate a number of ways to incorporate pragmatic reasoning into a CNN-LSTM architecture for REG, and also collected two datasets of referring expressions for objects in the MSCOCO corpus in the ReferItGame setup (Kazemzadeh et al., 2014). All these authors state that they use beam search during decoding, but do not investigate the effect of this search method (nor do they state exactly which variant of beam search was used). Our work suggests that the effectiveness of contextual features interacts with the decoding procedure.

2.3 Decoding for neural MT

We now turn to (neural) MT, where decoding algorithms for sequence generation have been investigated in detail. Here, beam search is the standard method for syntax- and phrase-based models (Rush et al., 2013), as well as for neural encoder-decoders (Freitag and Al-Onaizan, 2017). However, an important difference between the two is that candidates in phrase-based MT are completed in the same number of steps, whereas neural models generate hypotheses of different length and are biased for shorter output (Huang et al., 2017). To counteract this bias, OpenNMT (Klein et al., 2017) adopts three metrics for normalizing the coverage, length and end of sentence of candidate translations. Unfortunately, two of these metrics (coverage and end of sentence normalization) are based on the length of the source sentence, which is not available in REG. Another common NMT framework (Bahdanau et al., 2014) uses a shrinking beam where beam size is reduced each time a completed hypothesis is found, and search terminates when the beam size has reached 0.

Another shortcoming of beam search observed in previous work is that the beam tends to contain many candidates that share the same (most likely) prefix (Freitag and Al-Onaizan, 2017). This means that a relatively high value for beam size is needed to ensure that more diverse hypotheses that could potentially lead to more probable output are not excluded too early. A range of works have looked at modifying the objective of beam search such that more diverse candidates are considered during search (Li et al., 2016; Freitag and Al-Onaizan, 2017). See Section 4 for an overview of the beam search heuristics we use in this paper.

To overcome the limitations of heuristically defined beam search, Gu et al. (2017) introduce the notion of trainable decoding that takes a pretrained neural MT system and optimizes the decoder for any objective. They treat the decoder as a small actor network that learns to manipulate the hidden state of the underlying trained MT system. Whereas Gu et al. (2017) train the decoder actor network with a policy gradient method, Chen et al. (2018) present a supervised method to train the decoder. We will follow the latter approach, as described in Section 4.

3 REG Models

As the focus of this work is on the decoding step for neural REG models, we adopt a standard model architecture in language and vision: we use the pre-softmax layer of a pre-trained CNN to represent the image and the image region that a given target expression refers to. We train a standard LSTM to generate a word at each time step, conditioned on the visual vector and the previous words. Our LSTM mainly follows the implementation of (Xu et al., 2015)[1], one of the most widely known models for image captioning, but does not use attention over the image.

We note that Xu et al. (2015) use a deep decoding layer that computes the output word probability given the LSTM state, the context vector and the previous word including dropout and nonlinear (tanh) activation functions, similar to Yu et al. (2017) who also use dropout in the decoding layer, but only linear activation functions. Mao et al. (2015), on the other hand, adopt a simple linear layer for decoding the LSTM. In the following, we will investigate how these modeling decisions affect the performance, and how they interact with different search methods during inference.[2] We distinguish two variants of our model according to their decoding layer:

Linear decoding layer:

$$p = softmax(W_h h + b_h) \tag{1}$$

where p is the distribution over the vocabulary, W_h is the weight matrix, b_h is the bias term, and h is the hidden state of the LSTM.

[1] https://github.com/yunjey/show-attend-and-tell

[2] Please note that there are two aspects of decoding in our set-up: the decoding layer of the LSTM, and the decoding inference procedure applied during testing.

Deep decoding layer:

$$\begin{aligned} h_1 &= dropout(h_0) \\ h_2 &= tanh((W_h h_1 + b_h) + (W_z z + b_z) + x_{prev}) \\ h_3 &= dropout(h_2) \\ p &= softmax(W_{out} h_3 + b_{out}) \end{aligned} \tag{2}$$

where h_0 is the hidden state, z is the visual vector and x_{prev} the embedding of the previous word.

Finally, we also vary the input visual representation that the LSTM is conditioned on. Whereas Mao et al. (2015) extract visual representations of the region representing the target referent and the global image, Yu et al. (2016) report a slightly detrimental effect of including these global context features. Thus, we distinguish two variants of the model according to its visual representation:

Target: 4103-dimensional vector, obtained by cropping the image to the target region, resizing to 224 × 224, extracting its CNN pre-softmax features with VGG19 (Simonyan and Zisserman, 2014) and concatenating 7 spatial features of the region (see Schlangen et al. (2016) for these)

Global+target: 8119-dimensional vector, obtained by extracting the CNN pre-softmax features with VGG19 (Simonyan and Zisserman, 2014) for the entire image, and concatenating it with **target-only**

Training We set the word embedding layer size to 512, and the hidden state to 1024. We optimized with ADAM (with $\alpha = 0.001$), and the batch size set to 50. The word embedding layer is initialized with random weights. The number of training epochs was tuned for each model on the validation set.

4 Decoding Strategies

We now explain the different decoding strategies that will be combined with the REG models.

4.1 Greedy decoding

This is the simplest way to apply an LSTM based generator for testing: at each time step, the most likely word is selected and added to the generation history. Greedy decoding terminates when the end symbol is generated.

4.2 Beam search

Beam search generalizes greedy decoding and keeps a fixed number K of generation candidates that are expanded at each time step (greedy decoding corresponds to beam search with $K = 1$). This is computationally less efficient than greedy search, but often yields better results in NLP (see Section 2). A definition of a standard beam search algorithm that a lot of previous work has followed can be found in Graves (2012).

As discussed in Section 2.3, the general skeleton of the beam search algorithm allows for a number of modifications, that concern the following criteria: (i) pruning: which candidates are added to the beam for the next time step, (ii) termination: when does search stop, (iii) normalization: how to treat candidates of different length, (iv) beam size: constant or dynamic value for K. The summary of search strategies we will test is shown in Table 1.

In the simple beam search version, the decoder checks after each iteration, whether the top-most candidate is complete. If this is not the case, all current candidates remain for the next iteration, including complete hypotheses that have a rank > 1. During development, we noticed that this causes short, lower ranked complete hypotheses to climb up the beam too quickly and win over long hypotheses (see results in Section 6.3). Therefore, we also introduce a new variation of simple beam, called **same-len** in Table 1.The **same-len** variant excludes complete hypotheses on the beam that have a rank > 1, whereas in the standard version these would remain on the beam. In the **same-len** version a complete hypothesis either wins (when it is on top) or is pruned, which means that all candidates on the beam are of the same length (when counting the end symbol).

When candidates of different length are kept on the beam, we can normalize according to the following scores taken from Klein et al. (2017):

$$lp(y) = \frac{(5 + |Y|)^\alpha}{(5 + 1)^\alpha} \tag{3}$$

where $|Y|$ is the length of the candidate output and α is usually set to a value between 0.6 and 0.8 (Wu et al., 2016). We could not find an explanation for the constant being set to 5 in the literature. This length penalty is then used to boost probabilities of longer sequences in the following way:

$$score(y, x) = \frac{logP(y|x)}{lp(y)} \tag{4}$$

where $P(y|x)$ corresponds to the probability assigned to the candidate y at the given time step.

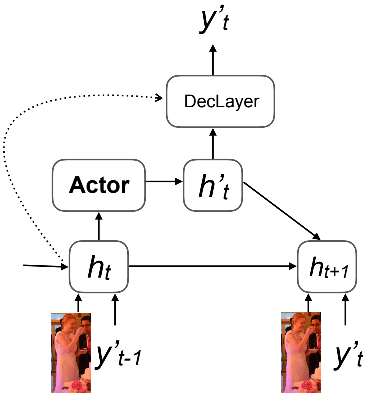


Figure 2: Illustration of neural REG architecture including an actor network that manipulates the hidden state; the dotted line indicates the information flow in the standard model where the hidden state is directly passed to the decoding layer

4.3 Trainable decoding

Finally, we set up a trainable decoder as in Chen et al. (2018) that adds a new *actor* module to the trained REG model. As illustrated in Figure 2, the actor learns to manipulate the hidden state that is then passed to the decoding layer. When training the actor, the weights of the underlying REG system are fixed. Thus, instead of optimizing the entire architecture in one stage, we adopt a procedure that first learns an REG model as usual, and then learns to decode this model in a second step.

For training the decoding actor in a supervised way, we need an 'artificial' corpus of referring expressions paired with objects in images that are both considered to be likely by the model and achieve a good BLEU score when compared to the original human expression. Thus, for an object x we use the REG model and beam search to produce a set of referring expressions $Y = y_1, y_2, ..., y_n$ that correspond to all completed hypotheses from all time steps, with a certain beam size K. But in contrast to the methods explained in Section 4.2, we do not need to define a heuristic stopping criterion. Instead, we use an oracle that selects y from the set Y that achieves the highest BLEU score given the human references. Of course, this oracle is only needed at training time. The actor itself is a simple feedforward layer that updates the hidden state as follows:

$$h' = h + relu(W_h h) \tag{5}$$

This is a slightly simpler definition than the one used in Chen et al. (2018) where the actor is also conditioned on the original input vector. We experimented with their version, but found a simple layer with ReLU activation to yield the best performance. When applying the actor at test time, it is possible to combine it with beam search (Gu et al., 2017). However, we only use it in a greedy fashion, as beam search did not lead to clear improvements in our experiments.

Training The main parameters to configure for training concern the definition of the oracle for compiling the new training data. We set K (beam size of the oracle) to 10, and use BLEU$_1$ (restricted to unigrams) as objective. We train the actor for 10 epochs, with ADAM.

5 Data

We conduct experiments on the RefCOCO(+) datasets, same as (Yu et al., 2016), which contain referring expressions to objects in MSCOCO (Lin et al., 2014) images. The data was collected via crowdsourcing with the ReferIt Game (Kazemzadeh et al., 2014) where two players were paired and a director needed to refer to a predetermined object to a matcher, who then selected it. Note that (Mao et al., 2015) performed experiments on a different data set for MSCOCO images in a non-interactive set-up. Thus, our evaluation set-up largely follows Yu et al. (2016, 2017).

RefCOCO and RefCOCO+ contain 3 referring expressions on average per object, and overall 150K expressions for 50K objects. The two datasets have been collected for an (almost) identical set of objects, but in RefCOCO+, players were asked not to use location words (*on the left*, etc.). See Yu et al. (2016) for more details. We use the predefined training and test splits. The respective test sets are divided into two subsets: **testA** is restricted to objects of the category human, **testB** consists of all other object types.

6 Experiments

6.1 Evaluation

Since we compare a whole range of REG models and decoding strategies, we opt for automatic evaluation measures, even though these might not fully reflect the performance that would be achieved in interaction with human users, cf. (Zarrieß and Schlangen, 2016). Unfortunately, different measures have been used in Yu et al. (2016) (BLEU$_1$, Meteor, ROUGE), and (Yu et al., 2017) (CIDEr, Meteor), which makes comparison less straightforward. Also note that Yu et al. (2017) state that they collected additional expressions for the test-sets, resulting in 10 expressions per objects. As we did not have access to these additional expressions at the time of writing, we follow Yu et al. (2016) and evaluate on the original RefCOCO collections with 3 expressions on average per object.

In the experiments below, we look at three measures: **BLEU$_1$** for unigrams (Papineni et al., 2002), **CIDEr** (Vedantam et al., 2015) and **lenr** (length ratio) as provided by the MSCOCO evaluation server (Chen et al., 2015). We are interested in the length ratio as a simple approximation of traditionally used measures in REG (Gatt and Belz, 2010), reflecting whether the generation output contains too much or too little information (attributes or words). **BLEU$_1$** gives us an indication of the lexical overlap between output and target, whereas **CIDEr** operates on the level of n-grams.

6.2 Simple beam search

In Table 2, we present results for the different configurations of our REG model, tested with greedy decoding and the simple variant of beam search, with $K = 3$ and $K = 10$. Our results for greedy decoding are comparable in BLEU$_1$ to the baseline results reported by Yu et al. (2016), with lower performance on RefCOCO+. This is likely due to different hyperparameter settings. The model comparison also shows that the combination of global context features and a deep decoding layer is clearly disadvantageous, especially for RefCOCO+. However, for the region model, the linear decoder outperforms the deep one on RefCOCO+, but underperforms it on RefCOCO.

When analyzing the effect of beam size on performance for 16 model-test set combinations in Table 2, some clear patterns can be observed:

- a wider beam always leads to shorter expressions as shown be the average length ratio, the effect is drastic on the RefCOCO+ data (e.g. the av. ratio on testB is 0.92 for greedy decoding and 0.45 for beam search with $K = 3$)
- a wider beam leads to lower BLEU$_1$ scores for most models and test sets, except for 1 out of 16 model-test set combinations
- a wider beam sometimes leads to better CIDEr scores, BLEU$_1$ and CIDEr disagree in 7 out of 16 model-test set combinations

These results clearly suggest that the stopping criterion defined in standard beam search is not appropriate for neural REG models. The greedy decoding can estimate the appropriate expression length surprisingly well, even in the region model that does not have access to global context. In contrast, the standard beam search that keeps candidates of different length seems clearly biased towards output that is too short. However, the fact that CIDEr scores still improve in some cases suggests that beam search leads to linguistically more well-formed expressions (expressions with a lot of repetitions are avoided, e.g. *the blue blue shirt*).

6.3 Modified beam search

In Table 3, we report performance of the region and global model with the linear decoder and investigate the effect of different beam search variants on performance (for reasons of space, we omit the models with a deep decoder, as these were more unstable on RefCOCO+).

- a shrinking beam has a clearly detrimental effect on performance in all cases
- the other beam search variants consistently improve over simple beam search on all metrics
- there is no clear improvement of the beam search variants over greedy decoding
- len-norm and same-len achieve better length ratios than simple beam, but not better than greedy decoding

These results support our initial hypothesis that knowing when to terminate is an essential aspect of REG, and this aspect is learnt relatively well by the LSTM. Beam search heuristics for termination seem to override the model's knowledge of what a good stopping point is. The heuristics for length normalization and stopping have some positive effect over simple beam search, but are not fully effective, and might need more extensive tuning. But overall, this suggests that a more principled decoding solution for neural REG is needed as greedy decoding also leads to undesired output patterns (repeated words, for instance).

6.4 Trainable decoding

Table 4 compares the results for the greedy decoder against a decoder trained to optimize BLEU$_1$ scores, as explained in Section 4. Some interesting observations can be made:

- the trained decoder improves over the greedy decoder on all test sets, models and measures

method	filtering	termination	normalization	beam size
simple	–	when top y is complete	–	constant
len-norm	–	when top y is complete	length (eq. 3, 4)	constant
same-len	discard y if complete, but not top	when top y is complete	–	constant
pruning (Freitag and Al-Onaizan, 2017)	discard y if complete, but not top; discard y if m candidates with same history are in beam	when top y is complete	–	constant
shrinking (Bahdanau et al., 2014)	–	when beam size is 0, select top y that is complete	length (eq. 3, 4)	-1 for each complete y

Table 1: Beam search variants, y refers to generation candidates

		testA			testB			testA+			testB+		
model	K	Bleu1	CIDEr	lenr	Bleu1	CIDEr	lenr	Bleu1	CIDEr	lenr	Bleu1	CIDEr	lenr
Yu et al. (2016)'s BL	-	0.477	-	-	0.553	-	-	0.391	-	-	0.331	-	-
target,deep	1	**0.484**	**0.625**	**0.89**	**0.516**	**1.096**	**0.77**	**0.361**	**0.436**	**1.04**	**0.264**	0.552	**0.85**
target,deep	3	0.452	0.604	0.77	0.471	1.088	0.63	0.304	0.412	0.67	0.180	**0.587**	0.43
target,deep	10	0.430	0.592	0.70	0.454	1.093	0.60	0.208	0.379	0.50	0.148	0.582	0.38
target,lin	1	**0.445**	0.555	**0.85**	**0.513**	**1.078**	**0.85**	**0.378**	**0.443**	**1.00**	**0.296**	0.506	**0.99**
target,lin	3	0.420	**0.568**	0.70	0.480	**1.077**	0.69	0.302	0.415	0.65	0.252	**0.604**	0.57
target,lin	10	0.369	0.535	0.62	0.464	1.065	0.66	0.222	0.393	0.51	0.212	0.601	0.48
target+global,deep	1	0.400	**0.602**	**1.12**	**0.464**	1.011	**0.86**	**0.329**	**0.387**	**1.04**	**0.231**	0.525	**0.95**
target+global,deep	3	**0.440**	0.597	0.72	0.399	**1.014**	0.56	0.231	0.337	0.57	0.140	**0.567**	0.36
target+global,deep	10	0.387	0.569	0.62	0.371	1.003	0.53	0.147	0.326	0.41	0.150	0.576	0.37
target+global,lin	1	**0.461**	**0.593**	**0.89**	**0.497**	**1.043**	**0.78**	**0.357**	**0.382**	**0.91**	**0.281**	0.558	**0.92**
target+global,lin	3	0.426	0.588	0.72	0.443	**1.046**	0.61	0.267	0.350	0.61	0.192	**0.590**	0.45
target+global,lin	10	0.370	0.546	0.61	0.420	1.028	0.58	0.186	0.327	0.47	0.154	0.586	0.39

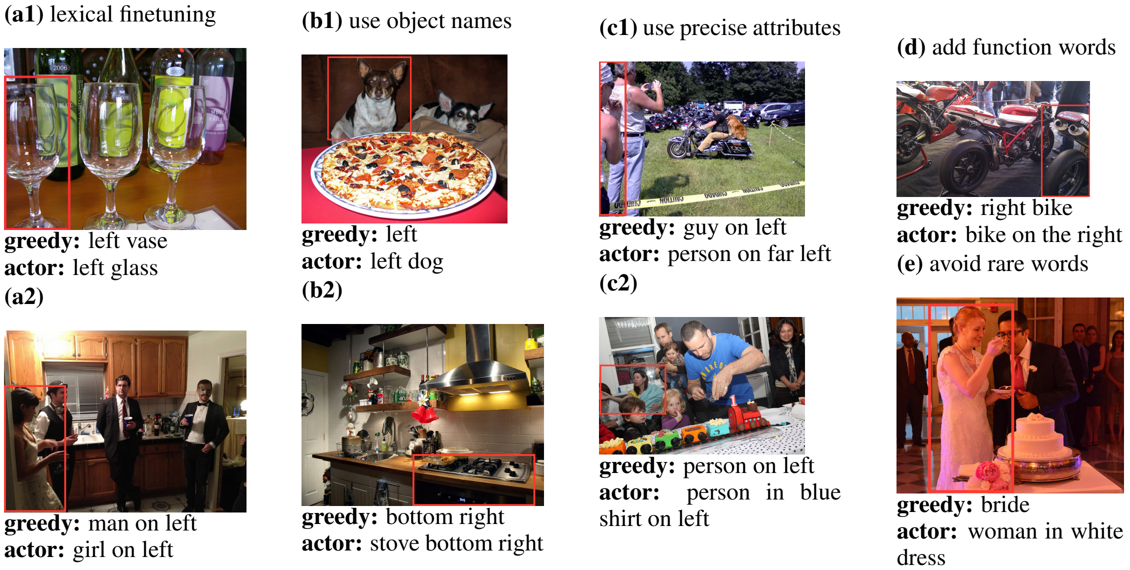
Table 2: Effect of beam size K on generation performance (beam with $K = 1$ corresponds to greedy decoding); comparing models with region and global features, and a deep vs. linear decoding layer.

		testA			testB			testA+			testB+		
model	decoder,K	Bleu1	CIDEr	lenr	Bleu1	CIDEr	lenr	Bleu1	CIDEr	lenr	Bleu1	CIDEr	lenr
target	greedy,1	0.445	0.555	**0.85**	**0.513**	1.078	**0.85**	**0.378**	**0.443**	**1.00**	**0.296**	0.506	**0.99**
target	simple,3	0.420	0.568	0.70	0.480	**1.077**	0.69	0.302	0.415	0.65	0.252	**0.604**	0.57
target	len-norm,3	0.454	0.598	0.77	0.504	**1.097**	0.75	0.314	0.417	0.71	0.275	0.616	0.64
target	len-norm,10	0.444	0.595	0.74	0.502	1.090	0.75	0.279	0.409	0.65	0.275	**0.622**	0.64
target	same-len,3	**0.464**	**0.601**	0.81	0.507	1.100	0.77	0.337	0.417	0.82	0.294	0.615	0.72
target	same-len,10	0.451	0.598	0.77	0.505	1.092	0.75	0.302	0.403	0.73	0.283	0.624	0.68
target	pruning,10	0.461	**0.602**	0.80	0.506	1.099	0.76	0.324	0.410	0.80	0.285	0.610	0.70
target	shrinking,3	0.237	0.317	2.48	0.272	0.600	2.06	0.254	0.319	1.45	0.230	0.460	1.34
target	shrinking,10	0.361	0.462	1.11	0.400	0.792	1.19	0.299	0.382	0.81	0.277	0.501	1.03
target+global	greedy,1	**0.461**	0.593	**0.89**	**0.497**	1.043	0.78	**0.357**	**0.382**	**0.91**	**0.281**	0.558	**0.92**
target+global	simple,3	0.426	0.588	0.72	0.443	**1.046**	0.61	0.267	0.350	0.61	0.192	**0.590**	0.45
target+global	len-norm,3	0.456	**0.613**	0.79	0.480	1.073	0.68	0.282	0.359	0.64	0.221	0.589	0.51
target+global	len-norm,10	0.441	0.601	0.76	0.484	1.073	0.69	0.237	0.355	0.57	0.218	**0.593**	0.50
target+global	same-len,3	**0.462**	**0.616**	0.84	0.490	**1.079**	0.71	0.328	0.380	0.79	0.264	0.588	0.64
target+global	same-len,10	0.447	0.606	0.79	0.490	1.072	0.71	0.269	0.357	0.66	0.255	0.582	0.60
target+global	pruning,10	0.459	0.617	0.83	0.488	1.068	0.70	0.302	0.366	0.74	0.258	0.588	0.61
target+global	shrinking,3	0.240	0.368	2.21	0.282	0.731	1.73	0.179	0.226	1.89	0.167	0.429	1.79
target+global	shrinking,10	0.379	0.469	0.89	0.436	0.843	**1.00**	0.275	0.314	0.75	0.233	0.545	0.62

Table 3: Model with linear decoding layer, different ways of normalizing/parametrizing beam search

		testA			testB			testA+			testB+		
model	decoder	Bleu1	CIDEr	lenr	Bleu1	CIDEr	lenr	Bleu1	CIDEr	lenr	Bleu1	CIDEr	lenr
target,deep	greedy	0.484	0.625	0.89	0.516	1.096	0.77	0.361	**0.436**	1.04	0.264	**0.552**	0.85
target,deep	bleu-actor	**0.507**	**0.658**	**0.95**	**0.534**	**1.112**	**0.82**	**0.377**	0.4	**1.02**	**0.269**	0.527	**0.75**
target,lin	greedy	0.445	0.555	0.85	0.513	1.078	0.85	**0.378**	0.443	**1.00**	0.296	0.506	**0.99**
target,lin	bleu-actor	**0.487**	**0.625**	**0.99**	**0.535**	**1.089**	**0.94**	0.377	**0.452**	1.09	**0.320**	**0.579**	0.97
target+global,lin	greedy	0.461	0.593	0.89	0.497	1.043	0.78	0.357	0.382	0.91	0.281	0.558	0.92
target+global,lin	bleu-actor	**0.498**	**0.655**	**1.02**	**0.549**	**1.083**	**0.93**	**0.375**	**0.419**	**1.02**	**0.315**	**0.587**	**0.93**

Table 4: Results for trainable decoding (actor model)



(a1) lexical finetuning

greedy: left vase
actor: left glass

(a2)

greedy: man on left
actor: girl on left

(b1) use object names

greedy: left
actor: left dog

(b2)

greedy: bottom right
actor: stove bottom right

(c1) use precise attributes

greedy: guy on left
actor: person on far left

(c2)

greedy: person on left
actor: person in blue shirt on left

(d) add function words

greedy: right bike
actor: bike on the right

(e) avoid rare words

greedy: bride
actor: woman in white dress

Figure 3: Examples from the RefCOCO validation set for strategies learned by the trained decoder

with only few exceptions (CIDEr scores on testA+ and testB+ for the deep region model)

- the improvements are substantial in many cases (between 2 and 8 points for BLEU$_1$, and up to 7 points for CIDEr)
- given the trained decoder, the global model now improves over the region model in the case of testB (BLEU, CIDEr, lenr) and testB+ (CIDEr)

These results demonstrate the importance of applying the right decoding procedure when generating with neural REG models. Interestingly, the qualitative improvements in performance we obtain clearly exceed the effects found with the same methods for MT (Chen et al., 2018), but generally support previous findings on positive effects when optimizing a sequence model for external evaluation metrics (Ranzato et al., 2016). A possible explanation is that our REG models benefit from a two-stage optimization procedure, similar to curriculum learning (Bengio et al., 2009). Another important question is whether BLEU$_1$, the objective we used to train the decoder, is conceptually appropriate for REG or whether we simply tune the model to our final evaluation measure. The fact that the actor model also improves the CIDEr scores and length ratios is a first positive indication that the BLEU$_1$-actor does not just fit to the metric in a superficial way. The qualitative analysis in Section 6.6 will shed more light on this.

6.5 Global context

Besides the effect of different decoding strategies on the performance of our neural REG model, an interesting and somewhat counterintuitive observation is that the models that incorporate global context features do not generally outperform the local models which only 'look' at the target referent. This finding seems to contradict some very basic assumptions that have been formulated in the REG literature, namely that the content of a referring expression (e.g. its attributes) depend on the distractors, cf. (Krahmer and Van Deemter, 2012). At the same time, a lot of theoretical and computational work on referring expressions has observed that human speakers tend to overspecify, i.e. use attributes even though they are not strictly needed to discriminate the target referent from its distractors (Koolen et al., 2011). Moreover, our findings seem to corroborate previous work on RefCOCO that even observed a detrimental effect of including global context features in a neural REG model.

Unfortunately, the RefCOCO corpora lack a ground truth annotation for attributes, hence, is it is hard to analyze whether the (missing) effect of global context is due to shortcomings of existing neural models for REG or due to inherent patterns in the data (such as e.g. overspecification). We believe that a more systematic approach to assessing the effect of distractors on content of referring expressions in real-world image corpora is a very promising direction for future research.

6.6 Analysis: Strategies learned via BLEU

We manually go through examples in the RefCOCO validation set, and broadly categorize the cases where the actor model improves over the greedy decoder. Figure 3 illustrates some frequent patterns we discovered with representative examples for each. Our analysis suggests that the actor has indeed automatically discovered strategies that lead to contextually more appropriate expressions: it learns to (a) fine-tune its lexicon and use object names in a sematically more adequate way, (b) include object names more often making expressions more pragmatically adequate, and (c) use more precise attributes also leading to more pragmatically adequate output. At the same time, the decoder also learns a strategy that can be considered as 'metric fitting', namely to (d) use more function words (articles, prepositions) which is a rather cheap strategy to increase BLEU scores. Finally, we find that the actor sometimes prevents the model from using short expressions containing rare words, as e.g. 'bride' in Figure 3(e).

These findings support the interpretation that a two-stage optimization set-up can help an REG model to pragmatically fine-tune its generation output. Vedantam et al. (2017) have recently adopted a similar approach, tuning a context-agnostic captioning system to produce discriminative captions at the stage of decoding (Yu et al., 2016).

7 Conclusion

We have investigated decoding strategies for neural REG, finding a clear advantage of trainable decoding optimized for BLEU over standard beam search methods. We think that this two-stage optimization set-up offers interesting directions for future work and can possibly be applied, for instance, in interactive learning scenarios and be tuned to more explicit communicative objectives.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Yun Chen, Kyunghyun Cho, Samuel R Bowman, and Victor OK Li. 2018. Stable and effective trainable greedy decoding for sequence to sequence learning. In *ICLR 2018, Workshop Track*, Vancouver, Canada.

Alasdair DF Clarke, Micha Elsner, and Hannah Rohde. 2013. Where's wally: the influence of visual salience on referring expression generation. *Frontiers in psychology*, 4.

Robert Dale and Ehud Reiter. 1995. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263.

Kees van Deemter, Albert Gatt, Ielka van der Sluis, and Richard Power. 2012. Generation of referring expressions: Assessing the incremental algorithm. *Cognitive Science*, 36(5):799–836.

Markus Freitag and Yaser Al-Onaizan. 2017. Beam search strategies for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 56–60, Vancouver. Association for Computational Linguistics.

Albert Gatt and Anja Belz. 2010. Introducing shared tasks to nlg: The tuna shared task evaluation challenges. In *Empirical methods in natural language generation*, pages 264–293. Springer.

Dimitra Gkatzia, Verena Rieser, Phil Bartie, and William Mackaness. 2015. From the virtual to the real world: Referring to objects in real-world spatial scenes. In *Proceedings of EMNLP 2015*. Association for Computational Linguistics.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. In *Representation Learning Worksop, ICML 2012*, Edinburgh, Scotland.

Jiatao Gu, Kyunghyun Cho, and Victor O.K. Li. 2017. Trainable greedy decoding for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1968–1978, Copenhagen, Denmark. Association for Computational Linguistics.

Liang Huang, Kai Zhao, and Mingbo Ma. 2017. When to finish? optimal beam search for neural text generation (modulo beam size). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2134–2139, Copenhagen, Denmark. Association for Computational Linguistics.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara L Berg. 2014. ReferItGame: Referring to Objects in Photographs of Natural Scenes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 787–798, Doha, Qatar.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Ruud Koolen, Albert Gatt, Martijn Goudbeek, and Emiel Krahmer. 2011. Factors causing overspecification in definite descriptions. *Journal of Pragmatics*, 43(13):3231–3250.

Emiel Krahmer, Sebastiaan van Erk, and André Verleg. 2003. Graph-based generation of referring expressions. *Computational Linguistics*, 29(1):53–72.

Emiel Krahmer and Kees Van Deemter. 2012. Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. A simple, fast diverse decoding algorithm for neural generation. *CoRR*, abs/1611.08562.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollr, and C.Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision ECCV 2014*, volume 8693, pages 740–755. Springer International Publishing.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L. Yuille, and Kevin Murphy. 2015. Generation and comprehension of unambiguous object descriptions. *ArXiv / CoRR*, abs/1511.02283.

Margaret Mitchell, Kees van Deemter, and Ehud Reiter. 2010. Natural reference to objects in a visual domain. In *Proceedings of the 6th international natural language generation conference*, pages 95–104. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *Proceedings of ICLR*.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge university press.

Alexander Rush, Yin-Wen Chang, and Michael Collins. 2013. Optimal beam search for machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 210–221.

David Schlangen, Sina Zarriess, and Casey Kennington. 2016. Resolving references to objects in photographs using the words-as-classifiers model. In *Proceedings of the 54rd Annual Meeting of the Association for Computational Linguistics (ACL 2016)*.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Ramakrishna Vedantam, Samy Bengio, Kevin Murphy, Devi Parikh, and Gal Chechik. 2017. Context-aware captions from context-agnostic supervision. In *Computer Vision and Pattern Recognition (CVPR)*, volume 3.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

Licheng Yu, Patrick Poirson, Shan Yang, Alexander C. Berg, and Tamara L. Berg. 2016. *Modeling Context in Referring Expressions*. Springer International Publishing, Cham.

Licheng Yu, Hao Tan, Mohit Bansal, and Tamara L Berg. 2017. A joint speakerlistener-reinforcer model for referring expressions. In *Computer Vision and Pattern Recognition (CVPR)*, volume 2.

Sina Zarrieß and David Schlangen. 2016. Easy things first: Installments improve referring expression generation for objects in photographs. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 610–620, Berlin, Germany. Association for Computational Linguistics.