The Syntactic Mechanism Behind Image Captioning

Meng Song
Carnegie Mellon University
songmeng86@hotmail.com

Xinlei Chen
Carnegie Mellon University
xinleic@cs.cmu.edu

Abstract

Driven by the success of deep neural network in machine translation and high-level visual feature extraction, image captioning has attracted significant attention in the recent year. To address this challenging problem, the encoder-decoder architecture is widely adopted, where RNNs are used as language model to decode the visual content into a sequence of words. These sequential neural models are considered to be able to implicitly learn the sentence structure. However, we lose the interpretability of how the compositional structure of a sentence is expanded along with the generation of words. This paper aims to provide a hypothesis of the underlying syntactic mechanism by constructing a new tree-structured neural network, R-RNN. The model explicitly generates the grammar and shows how the grammar and semantic units influence each other through its connections. We validate this hypothesis with comparable performance of image captioning as the vanilla RNN model on MS COCO dataset.

1. Introduction

In recent years, the emergence of the encoder-decoder architecture has shown great success in machine translation [1] and image caption generation ([17], [4], [5], [18]. In this framework, the encoder network first transforms one representation, such as an image, or a source sentence, to a fixed length semantic vector, then the decoder network decodes the vector into a caption or a target sentence. The specific networks chosen as encoders and decoders highly rely on the structure of the input and the output modality. Typically, on the vision part, the convolutional neural networks are used to extract the semantic information by exploiting the 2D spatial structure of an image, while on the language part, the RNNs are employed to model the temporal structure of a sentence. However, this sequential structure reflects the characteristics of speech, not language. Directly translating our thoughts or mental states into words from left to right assumes that the content of a sentence is determined in a single direction. This flat model does not naturally align with the hierarchical structure of a sentence from the point of view of linguistics. A key piece that bridges the meaning in our minds and the words is missing. That is the grammar structure.

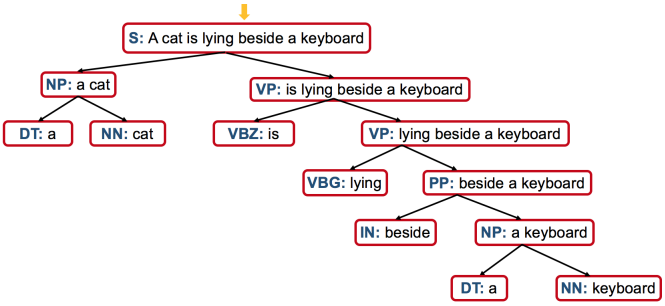

Figure 1. The hierarchy of sentence generation illustrated by a constituency tree. The grammar (non-leaf) nodes are associated with the relevant sentence fragments , and the leaf nodes indicate the words.

According to Chomsky Hierarchy [2], the sentences are generated subconsciously according to a set of formal grammar rules, which is called generative grammar. Following the rules, the mental representation, or the thought vector, is structured to a sentence in a recursive way from rough to fine. This means that on a constituency tree, the word is determined by the sequence of top-down grammar units governing it. Under this assumption, people embody their thoughts by gradually replacing the initial abstract ideas by segmental meanings and finally by specific words (Fig. 1).

Furthermore, we think that the words should also influence the grammar generation. We thus hypothesize that the procedure of people expressing their thoughts by a concrete sentence can be well described by the pre-order traversal of a constituency tree. This order is consistent with the way people say a sentence. As a result, every node on the tree, whatever it is grammar tag or word, is determined by both the grammar history along the path starting from the root and the word history beginning with the leftmost leaf node.

To validate our hypothesized mechanism of sentence generation, we proposed a novel neural network, called recursive recurrent neural network (R-RNN). The core of the

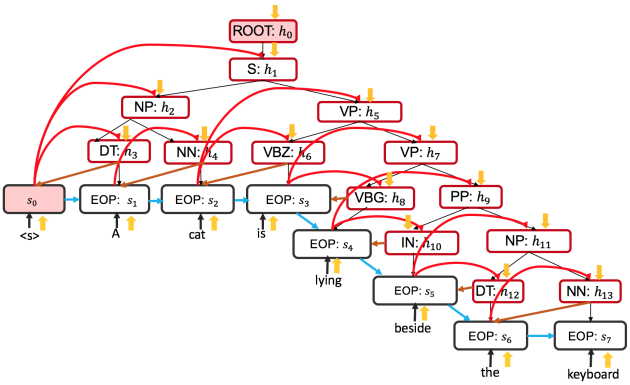

Figure 2. The architecture of R-RNN model.

R-RNN consists of two connected parts: a tree-structured grammar model, and a sequential word model. Each top-down path on the tree is modeled as a RNN with the grammar history shared at each parent node, while another horizontal RNN is built on the bottom to model the word sequence (Fig. 2). Then two groups of connections are added to interweave the left-right and top-down information flows. This architecture endows R-RNN with the capability of jointly generating the grammar and the words. More importantly, unlike most approaches which model the correlations of words as a single RNN, our model takes an inside look of how the language works when conveying the meaning. It explicitly points out the dependency of each grammar unit and word by its structure.

2. Related Work

Image Captioning Recently the encoder-decoder architecture has accelerated the pace of the community in exploring the topic of image captioning. Since [17] and [4] first introduced this CNN-RNN framework to the caption generation task, a lot of works have been done to improve the networks. [5] proposed a bi-directional RNN to extract the word representation in a wider context. [10] added multi-layers of word embeddings to help the multimodal embedding. In [3], a recurrent visual memory is proposed to learn the long-term dependencies. [18] creatively uses the attention mechanism to associate the image region to the generated word, thus visualizes the sentence generation process. However, almost all these works employ the sequential RNN to model the sentence, and few works pay attention to the inside of how the language works in the process of caption generation.

Grammar in Caption Generation [7] inputs the part-of-speech tags into the neural networks to provide extra context information. [8] uses a grammar template to composite captions for new images. However, none of them generates the grammar automatically.

Tree-structured Neural Networks A variety of recursive models [15], [14],[9] are proposed to learn compositional meaning representation of the sentence by traversing a dependency tree in a bottom-up order. Different from these approaches, we model the grammar generation or the decomposition of the sentence meaning in a top-down order.

Some efforts have been put on investigating the ability of RNN to model the complex structures in data. In [6], RNNs are used as character-level language models. [16], LSTMs with an attention model have shown to achieve the new state-of-the-art results in syntactic constituency parsing, but require a lot more training data. [12] further indicates that neural sequence models like RNNs have the capability of implicitly capturing the recursive compositional structures in the data. However, due to having the model structure as prior knowledge, tree-structured models can achieve the same performance more efficiently and with more transparency into the model.

3. Network Architecture

In this section, we first recall the basic RNN model, then we introduce our R-RNN model piece by piece in details.

3.1. Vanilla Recurrent Neural Network

The recurrent neural networks (RNNs) have long been used as neural language models. The goal of a statistical language model is to compute a probability distribution over a given sentence $S = \{x_1, ..., x_n\}$. Based on the chain rule, the distribution of a sequence of words can be factorized as

$$P(x_1, ..., x_n) = \prod_{t=1}^{N} P(x_t | x_1, ..., x_{t-1}) \quad (1)$$

RNN models the conditional probability $P(x_t|x_1, ..., x_{t-1})$ by summarizing the word history $x_1, ..., x_{t-1}$ into a fixed-length hidden state h_{t-1}. At each time step t, a new word x_t is added to update the history (Eq. 2). Then the whole history is used to predict the probability of the next word y_t (Eq. 3). In this first order dynamical system, the hidden state acts as a memory to keep track of the important aspects of past sequence to help the prediction.

$$h_t = \sigma(W^{hx} x_t + W^{hh} h_{t-1}) \quad (2)$$

$$y_t = \Phi(W^{yh} h_t) \quad (3)$$

Typically, word x_t is represented as a one-hot vector which is of the equal size as the vocabulary. After training, each column of the mapping matrix W^{hx} will correspond to the embedding of one word in the vocabulary. When it comes to the image captioning, the visual feature is added to Eq. 2

as a context vector. Then we can think of the word embeddings and the visual feature as existing in the same semantic space. They therefore pick the next word together.

Note that in this paper, for fair comparison, the logistic sigmoid function $\sigma(z) = 1/(1 + exp(-z))$ is used as the activation function $\sigma(\cdot)$ for all the models, the clipping is introduced to avoid the gradient explosion. $\Phi(\cdot)$ represents the softmax function.

3.2. Our R-RNN Model

Different from RNN, our recursive recurrent neural networks (R-RNNs) model the joint distribution of the sentence S and the grammar $P(S, G)$, where grammar G is represented as the constituency tree corresponding to S.

As illustrated in Fig. 2, nodes on the constituency tree fall into two categories: non-leaf nodes and leaves. Each non-leaf node is labeled by a non-terminal grammar tag, and is broken down into its child nodes according to a generating rule, e.g. $S \rightarrow NP, VP$. Each leaf is labeled with a special ending grammar symbol EOP as well as a word. Therefore, we model the generation of these two kinds of nodes in R-RNN respectively.

3.2.1 The Grammar Generation

For the grammar node visited at time step t, a modified RNN is introduced to model the generating rule $x_t \rightarrow \{g_1, ..., g_m\}_t$. Specifically, we want to achieve the branching by RNN's sequential prediction. To do so, unlike the classical RNN where the input symbol and output symbol share the same dictionary, we build one left-hand dictionary for inputs and one right-hand dictionary for outputs. The word in the right-hand dictionary is not one single symbol any more, but a sequence of symbols $y_t = \{g_1, ..., g_m\}_t$. This sequence decides the child nodes of node t from left to right, thus generates the structure of the tree. Fig 3 shows our special RNN by an example. At node t, the model first updates the grammar memory of its parent node h_{t-1} with input grammar x_t and produces new state h_t (Eq. 4). Then it uses h_t to predict its child sequence by selecting the most likely $\hat{y}_t$ from the right-hand dictionary, and branches according to either the predicted sequence or the groundtruth sequence. Note that at training time, the true right-hand sequence y_t is fed into the model for guiding the expansion, while at generating time, $\hat{y}_t$ is provided as the next input. The structure is thus decided by the predicted sequence $\hat{y}_t$. In our model, the hidden states of sibling nodes are all derived from their parents hidden state. This memory sharing is significantly different from modeling each root-to-leaf path by an independent RNN in [11].

Next, we let the word history influence the grammar generation by introducing word history s_k to Eq. 5. During the pre-order traversal, s_k is the word history up to the word

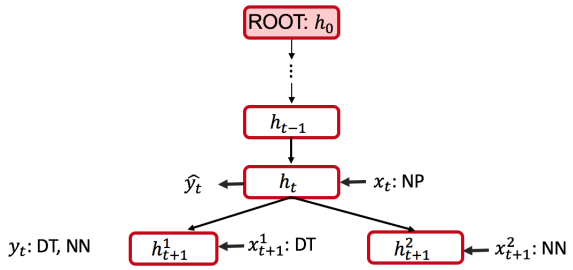

Figure 3. Model the generation of a grammar node without the word history.

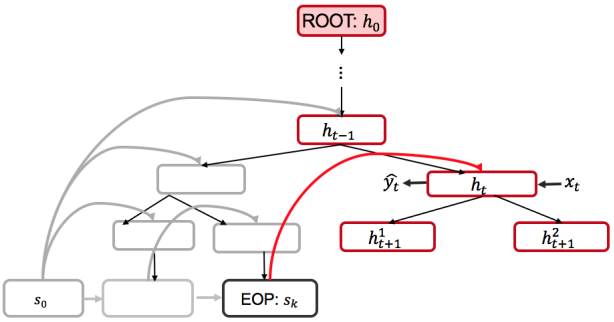

Figure 4. Model the generation of a grammar node with the word history.

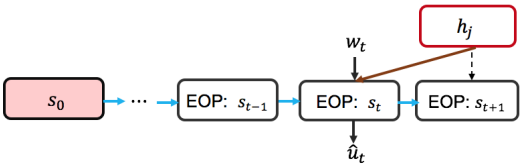

Figure 5. Model the generation of a word node.

right before node t (Fig. 4). The intuition is that the words have been spoken usually imply the next grammar unit.

In summary, our grammar generation model assumes that $P(x_t|G(x_t), S(x_t))$, the probability of grammar unit x_t, is conditioning on $G(x_t)$ and $S(x_t)$, where $G(x_t)$ is the sequence of grammar relations along the path starting from root to the parent of x_t , and $S(x_t)$ is the sequence of words spoken up to now.

$$h_t = \sigma(W_L x_t + W_H h_{t-1}) \quad (4)$$

$$y_t = \Phi(W_R h_t + W_V s_k) \quad (5)$$

3.2.2 The Sentence Generation

To generate a sentence, we model the leaves from left to right as a basic RNN with incorporating additional grammar history at each leaf. For leaf visited at time step t, the ground truth word w_t is given to update the word his-

tory s (Eq. 6). Then both s_t and the grammar history of next word h_j are used to decide the next word (Eq. 7). The modeled probability of word u_t has the same form $P(u_t|G(x_t), S(x_t))$ as the grammar unit in the grammar generation model.

$$s_t = \sigma(W_W w_t + W_S s_{t-1}) \quad (6)$$

$$u_t = \Phi(W_M h_j + W_U s_t) \quad (7)$$

4. R-RNN As Sentence Generator

In this section, we talk about training our model on the image captioning task and generating captions for new images. Different from other approaches trained on image and sentence pair, the training sample input into the R-RNN is a triplet (S, G, I) where I is an image, S is one ground truth caption of I, and G is the constituency tree of S.

4.1. Training

R-RNN is trained by using mini-batch stochastic gradient descent to maximize the posterior probability of the sentence and its grammar given the image w.r.t. the parameters θ for all training triplets (S, G, I).

$$\theta^* = \arg\max_{\theta} \sum_{(S,G,I)} \log P(S, G|I; \theta) + \frac{1}{2}\beta \left\|\theta\right\|_2^2 \quad (8)$$

Specifically, during the pre-order visit of the constituency tree, the model trying to predict each grammar tag on the tree and each word of the sentence, after it has seen the image at each time step. According to R-RNN, we have

$$\log P(S, G|I; \theta) = \sum_{x} \log P\Big(x|I, G(x), S(x); \theta\Big) + \sum_{u} \log P\Big(u|I, G(u), S(u); \theta\Big) \quad (9)$$

where x is any grammar tag on G, and u is the word.

By using grammar as additional signals, our model separates the semantic modeling and the syntactic modeling (Eq. 9). Therefore comparing to the RNN, our model provides a way to inspect the sentence generation process, and to tell whether the error is a grammatical one or a semantic one.

We pre-train the CNN on ImageNet 1000-way classification task, and fix the CNN part during the training of R-RNN. For each image, its fc-7 feature is fed at each grammar node and word node as extra input in Eq. 6 and Eq. 5. We use one sentence-tree-image triplet as a mini-batch to update the weights in R-RNN. All the weights are randomly initialized. In each iteration, two sentinel nodes are set up in advance to trigger the unrolling process. One is the root node which has the initial grammar memory h_0 and a special grammar starting symbol $ROOT$. The other one is an additional leaf node with the initial word memory s_0 and a special sentence starting symbol $\langle s \rangle$. We call this leaf l_0. The feedforward starts with the root node and follows the pre-order. When the first leaf l_1 is met, l_0 and grammar structure governing l_1 are used to predict the first word. Note that since the sentence structure is defined by the tree, we do not need EOS (End of Sentence) to control the sentence length as the typical RNN. The back-propagation is performed in the opposite direction of pre-order traversal.

4.2. Inference

Given an image, the natural way to generate a sentence from RNN is to greedily pick the most likely word at each step and feed it into the model at the next step. However, since during the inference we do not have the groundtruth to steer the model towards the correct direction, the discrepancy between the ground truth and the generated word will lead to errors accumulating quickly. To solve this problem, we need to introduce some diversity at each step. The most common way is to generate the sentence with beam search, which keeps more than one sentences at each time step as candidates to decide the next word. However, since our model generates the grammar structures automatically, different predictions of the right-hand sequences will correspond to different expansions of the tree. It therefore hard to compare the candidates using beam search. As an alternative solution, we simulate the beam search by randomly sampling from the top five candidates.

5. Experiments

5.1. Dataset

We conduct the experiments on the largest and the most cited dataset in image captioning, the MS COCO dataset. It contains 82,783 images for training and 40,504 images for validation, each with 5 groundtruth captions generated by human. The average length of the captions is 10.5 words per sentence. We train the RNN and R-RNN on the training set, but do not evaluate the models on the test set using MS COCO evaluation server. Instead, we follow the splits in [5] to select two subsets of 5,000 images from the validation set, and use one for validation, the other one for testing.

5.2. Implementation

We implement both of our R-RNN model and the vanilla RNN model from the scratch in C++. We optimize the computation using MKL BLAS and parallelize it using OpenMP. The models are trained with 8-core CPU.

To use the text in the dataset, we first preprocess each caption by trimming the unnecessary leading and tailing characters. Then we lowercase all the words and tokenize the sentences using Stanford CoreNLP Tool. To construct the word dictionary, all words occur less than 5 times are replaced by a special symbol OOV (out-of-vocabulary). The

size of word dictionary built on the COCO training set is 8,840.

Since our R-RNN model requires the grammar structure as extra inputs, we use the Stanford parser to parse each caption in the COCO dataset into a constituency tree. Note that since the parser is built on a statistical model, it will inevitably introduce mistakes into the constituency tree.

To construct the two grammar dictionaries, we extract each parent-child pair as a generation rule from the constituency trees of all the training captions, and remove the ones appear less than 5 times. By counting the distinct left-hand sides and right-hand sides of the rules, we obtain a left-hand dictionary with 65 grammar tags, and a right-hand dictionary including 1,744 grammar sequences.

During the training of R-RNN and RNN, we use early stop to avoid overfitting. For each epoch, we test the model on the validation set. If the current loss on validation set is larger than 99.7% of the previous loss, the performance improvement is considered to be not significant enough. We thus reduce the learning rate by half. The learning process stops when the learning rate has already dropped for 4 times.

In this paper, we use VGG-NET [13] to extract the visual features from the images due to its powerful generalization ability.

5.3. Model Comparison Results

To demonstrate the effectiveness of R-RNN as a language model and the necessity of some key designs in our model , we use the perplexity (PPL) as a metric to evaluate the variants of our model and the baseline model vanilla RNN on the COCO dataset. Note that when testing the models, instead of inputting words and tags, we feed the groundtruth into the model at each time step to enable the comparison.

On the sentence side, PPL is the probability of generating the testing sentence normalized by the number of words. The lower value indicates a better model. We compare RNN and our R-RNN to tell how much of the perplexity reduction comes from knowing the grammar information. For RNN, we compute PPL per word according to its standard definition, that is

$$PPL(S) = \Big(\prod_{t=1}^{N} P(x_t|x_1, ..., x_{t-1})\Big)^{-\frac{1}{N}} \quad (10)$$

where $P(x_t|x_1, ..., x_{t-1})$ is computed according to Eq. 3.

For R-RNN, we compute its PPL per word conditioned on the grammar tags along the root-to-leaf path as

$$PPL(S|G) = \Big(\prod_{t=1}^{N} P(x_t|S(x_t), G(x_t))\Big)^{-\frac{1}{N}} \quad (11)$$

	w/o visual feature	w visual feature
LR	2.86	-
R-RNN-g	2.38	2.29
R-RNN	2.11	2.07

Table 1. Perplexity per grammar tag on COCO test set. The memory size of RNN is 100. The word memory size and the grammar memory size of R-RNN are both set to 100.

where $P(x_t|S(x_t), G(x_t))$ is computed according to Eq. 7.

On the grammar side, the uncertainty of modeling the constituency tree of the testing sentence is measured on each root-to-leaf path. PPL per grammar is defined as the probability of generating the grammar tags on all of the root-to-leaf paths of the constituency tree normalized by the number of tags.

We first set up a baseline model (LR) which only uses the current grammar tag to predict the next one on the root-to-leaf path, and its PPL per grammar tag is computed as

$$P(u_t|u_{t-1}) = \Phi(u_{t-1}) \quad (12)$$

$$PPL(G) = \Big(\prod_{t=1}^{M} P(u_t|u_{t-1})\Big)^{-\frac{1}{M}} \quad (13)$$

Then we provide a variant of R-RNN to the comparison (R-RNN-g). We remove the word history $W_V s_k$ from Eq. 5. In this case, current grammar tag only relies on its ancestor grammar tags. Then PPL per grammar tag is computed as Eq. 14. Comparing LR and R-RNN-g tells us how much does the grammar history help the model improve the prediction accuracy.

$$PPL(G) = \Big(\prod_{t=1}^{M} P(u_t|u_1, ..., u_{t-1})\Big)^{-\frac{1}{M}} \quad (14)$$

Next, we show Eq. 15 to compute PPL per grammar for our R-RNN model, where $P(u_t|S(u_t), G(u_t))$ is defined in Eq. 5. By comparing R-RNN-g and R-RNN, we know how much uncertainty on grammar structure can be reduced by knowing the previous words in the sentence.

$$PPL(G|S) = \Big(\prod_{t=1}^{M} P(u_t|S(u_t), G(u_t))\Big)^{-\frac{1}{M}} \quad (15)$$

The comparison results are shown in Table 1 and 2 in two settings, with or without feeding the visual feature of the given image into the models. The difference of the same model's performances in two settings measures the information gain from the visual content of the image.

5.4. Image Caption Generation Results

In this section, we evaluate our R-RNN model's ability to generate the description of a new image on the COCO

	Bleu-1	Bleu-2	Bleu-3	Bleu-4	ROUGEL	METEOR	CIDEr
R-RNN	0.61	0.41	0.28	0.19	0.44	0.19	0.53
RNN	0.60	0.41	0.28	0.20	0.44	0.19	0.53

Table 3. Caption genration results on COCO test set. The memory size of RNN is 100. The word memory size and the grammar memory size of R-RNN are both set to 100.

	w/o visual feature	w visual feature
RNN	25.11	15.34
R-RNN	11.80	6.87

Table 2. Perplexity per word on COCO test set. The memory size of RNN is 100. The word memory size and the grammar memory size of R-RNN are both set to 100.

dataset, and compare it with the vanilla RNN. The generated captions are evaluated by the standard metrics for caption generation task, that is BLEU from 1-gram to 4-gram, METEOR and CIDEr. For all of these three metrics, higher values are better. To generate a sentence using R-RNN, we randomly pick the grammar tag from the top 5 choices, and choose the best word according to our model in a greedy way. This is because by randomly sampling the grammar structure, the word candidates already have the diversity in the part of speech. Randomly choosing words in a specific word class will only change the semantic meaning of the word, which instead reduces the prediction accuracy. For example, when the model predicts the word as a noun, although the top choices, such as cat and dog, are ranked closely, will significantly change the meaning of the sentence. The results in Table 3 demonstrate that our model learns to generate the grammar and the caption simultaneously with performance comparable to RNN. Note that our model is slightly better than RNN in Bleu-1 while RNN is better than our model in Blue-4. This is probably because by explicitly generating the grammar, our model provides more flexible structures when organizing the same key words.

6. Conclusions

In conclusion, we propose a hypothesis that people convey their thoughts by the hierarchy of grammar. In this process, the word history and grammar history together impact the word and grammar prediction in the order corresponding to the pre-order traversal of the constituency tree. To validate this hypothesis, we design a novel neural network R-RNN that explicitly embeds the proposed dependencies into its connections. The experimental results on the COCO dataset show that our R-RNN model has the same ability in caption generation as the vanilla RNN, while our model can also jointly learn the grammar generation. Through its architecture, our R-RNN model provides an possible explanation of how human form a sentence, and we therefore can take a look at the inside mechanism of how the RNN decodes the unified thought vectors, such as a visual feature or the encoding vector of a given sentence.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] A. Carnie. *Syntax: A generative introduction*. John Wiley and Sons, 2013.

[3] X. Chen and C. Lawrence Zitnick. Mind's eye: A recurrent visual representation for image caption generation. June 2015.

[4] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. *arXiv preprint arXiv:1411.4389*, 2014.

[5] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *CVPR*, 2015.

[6] A. Karpathy, J. Johnson, and F.-F. Li. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

[7] R. Kiros, R. Salakhutdinov, and R. S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*, 2014.

[8] P. Kuznetsova, V. Ordonez, T. L. Berg, and Y. Choi. Treetalk: Composition and compression of trees for image descriptions.

[9] M. Ma, L. Huang, B. Zhou, and B. Xiang. Tree-based convolution for sentence modeling. *arXiv preprint arXiv:1506.04834*, 2015.

[10] J. Mao, W. Xu, Y. Yang, J. Wang, and A. Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*, 2014.

[11] P. Mirowski and A. Vlachos. Dependency recurrent neural language models for sentence completion. *ACL*, 2015.

[12] C. P. Samuel R. Bowman, Christopher D. Manning. Tree-structured composition in neural networks without tree-structured architectures. *arXiv preprint arXiv:1507.01839*, 2015.

[13] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[14] R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218, 2014.

[15] K. S. Tai, R. Socher, and C. D. Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

[16] O. Vinyals, L. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton. Grammar as a foreign language. *arXiv preprint arXiv:1412.7449*, 2014.

[17] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition*, 2015.

[18] K. Xu, J. Ba, R. Kiros, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2015.