CENTER FOR
Brains
Minds+
Machines

CBMM Memo No. 082 **May 10, 2018**

Scene Graph Parsing as Dependency Parsing

Yu-Siang Wang[1], **Chenxi Liu**[2], **Xiaohui Zeng**[3], **Alan Yuille**[2]

1: National Taiwan University
2: Johns Hopkins University
3: Hong Kong University of Science and Technology

Abstract

In this paper, we study the problem of parsing structured knowledge graphs from textual descriptions. In particular, we consider the scene graph representation that considers objects together with their attributes and relations: this representation has been proved useful across a variety of vision and language applications. We begin by introducing an alternative but equivalent edge-centric view of scene graphs that connect to dependency parses. Together with a careful redesign of label and action space, we combine the two-stage pipeline used in prior work (generic dependency parsing followed by simple post-processing) into one, enabling end-to-end training. The scene graphs generated by our learned neural dependency parser achieve an F-score similarity of 49.67% to ground truth graphs on our evaluation set, surpassing best previous approaches by 5%. We further demonstrate the effectiveness of our learned parser on image retrieval applications.

This material is based upon work supported by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF-1231216.

Scene Graph Parsing as Dependency Parsing

Yu-Siang Wang
National Taiwan University
b03202047@ntu.edu.tw

Chenxi Liu(✉)
Johns Hopkins University
cxliu@jhu.edu

Xiaohui Zeng
Hong Kong University of Science and Technology
xzengaf@connect.ust.hk

Alan Yuille
Johns Hopkins University
alan.yuille@jhu.edu

Abstract

In this paper, we study the problem of parsing structured knowledge graphs from textual descriptions. In particular, we consider the scene graph representation (Johnson et al., 2015) that considers objects together with their attributes and relations: this representation has been proved useful across a variety of vision and language applications. We begin by introducing an alternative but equivalent edge-centric view of scene graphs that connect to dependency parses. Together with a careful redesign of label and action space, we combine the two-stage pipeline used in prior work (generic dependency parsing followed by simple post-processing) into one, enabling end-to-end training. The scene graphs generated by our learned neural dependency parser achieve an F-score similarity of 49.67% to ground truth graphs on our evaluation set, surpassing best previous approaches by 5%. We further demonstrate the effectiveness of our learned parser on image retrieval applications.[1]

1 Introduction

Recent years have witnessed the rise of interest in many tasks at the intersection of computer vision and natural language processing, including semantic image retrieval (Johnson et al., 2015; Vendrov et al., 2015), image captioning (Mao et al., 2014; Karpathy and Li, 2015; Donahue et al., 2015; Liu et al., 2017b), visual question answering (Antol et al., 2015; Zhu et al., 2016; Andreas et al., 2016), and referring expressions (Hu et al., 2016; Mao et al., 2016; Liu et al., 2017a). The pursuit for these tasks is in line with people's desire for high level understanding of visual content, in particular, using textual descriptions or questions to help understand or express images and scenes.

What is shared among all these tasks is the need for a *common representation* to establish connection between the two different modalities. The majority of recent works handle the vision side with convolutional neural networks, and the language side with recurrent neural networks (Hochreiter and Schmidhuber, 1997; Cho et al., 2014) or word embeddings (Mikolov et al., 2013; Pennington et al., 2014). In either case, neural networks map original sources into a semantically meaningful (Donahue et al., 2014; Mikolov et al., 2013) vector representation that can be aligned through end-to-end training (Frome et al., 2013). This suggests that the vector embedding space is an appropriate choice as the common representation connecting different modalities (see e.g. Kaiser et al. (2017)).

While the dense vector representation yields impressive performance, it has an unfortunate limitation of being less intuitive and hard to interpret. Scene graphs (Johnson et al., 2015), on the other hand, proposed a type of directed graph to encode information in terms of objects, attributes of objects, and relationships between objects (see Figure 1 for visualization). This is a more structured and explainable way of expressing the knowledge from either modality, and is able to serve as an alternative form of common representation. In fact, the value of scene graph representation has already been proven in a wide range of visual tasks, including semantic image retrieval (Johnson et al., 2015), caption quality evaluation (Anderson et al., 2016), etc. In this paper, we focus on scene graph generation from textual descriptions.

Previous attempts at this problem (Schuster et al., 2015; Anderson et al., 2016) follow the same spirit. They first use a dependency parser to obtain the dependency relationship for all words in a sentence, and then use either a rule-based or a learned classifier as post-processing to generate the scene graph. However, the rule-based classifier cannot

[1]Code is available at https://github.com/Yusics/bist-parser/tree/sgparser

Figure 1: Each image in the Visual Genome (Krishna et al., 2017) dataset contains tens of region descriptions and the region scene graphs associated with them. In this paper, we study how to generate high quality scene graphs (two such examples are shown in the figure) from textual descriptions, without using image information.

learn from data, and the learned classifier is rather simple with hand-engineered features. In addition, the dependency parser was trained on linguistics data to produce complete dependency trees, some parts of which may be redundant and hence confuse the scene graph generation process.

Therefore, our model abandons the two-stage pipeline, and uses a single, customized dependency parser instead. The customization is necessary for two reasons. First is the difference in label space. Standard dependency parsing has tens of edge labels to represent rich relationships between words in a sentence, but in scene graphs we are only interested in three types, namely objects, attributes, and relations. Second is whether every word needs a head. In some sense, the scene graph represents the "skeleton" of the sentence, which suggests that empty words are unlikely to be included in the scene graph. We argue that in scene graph generation, it is unnecessary to require a parent word for every single word.

We build our model on top of a neural dependency parser implementation (Kiperwasser and Goldberg, 2016) that is among the state-of-the-art. We show that our carefully customized dependency parser is able to generate high quality scene graphs by learning from data. Specifically, we use the Visual Genome dataset (Krishna et al., 2017), which provides rich amounts of region description - region graph pairs. We first align nodes in region graphs with words in the region descriptions using simple rules, and then use this alignment to train our customized dependency parser. We evaluate our parser by computing the F-score between the parsed scene graphs and ground truth scene graphs. We also apply our approach to image retrieval to show its effectiveness.

2 Related Works

2.1 Scene Graphs

The scene graph representation was proposed in Johnson et al. (2015) as a way to represent the rich, structured knowledge within an image. The nodes in a scene graph represent either an object, an attribute for an object, or a relationship between two objects. The edges depict the connection and association between two nodes. This representation is later adopted in the Visual Genome dataset (Krishna et al., 2017), where a large number of scene graphs are annotated through crowd-sourcing.

The scene graph representation has been proved useful in various problems including semantic image retrieval (Johnson et al., 2015), visual question answering (Teney et al., 2016), 3D scene synthesis (Chang et al., 2014), and visual relationship detection (Lu et al., 2016). Excluding Johnson et al. (2015) which used ground truth, scene graphs are obtained either from images (Dai et al., 2017; Xu et al., 2017; Li et al., 2017) or from textual descriptions (Schuster et al., 2015; Anderson et al., 2016). In this paper we focus on the latter.

In particular, parsed scene graphs are used in Schuster et al. (2015) for image retrieval. We show that with our more accurate scene graph parser, performance on this task can be further improved.

2.2 Parsing to Graph Representations

The goal of dependency parsing (Kübler et al., 2009) is to assign a parent word to every word in a sentence, and every such connection is associated with a label. Dependency parsing is particularly suitable for scene graph generation because it directly models the relationship between individual

words without introducing extra nonterminals. In fact, all previous work (Schuster et al., 2015; Anderson et al., 2016) on scene graph generation run dependency parsing on the textual description as a first step, followed by either heuristic rules or simple classifiers. Instead of running two separate stages, our work proposed to use a single dependency parser that is end-to-end. In other words, our customized dependency parser generates the scene graph in an online fashion as it reads the textual description once from left to right.

In recent years, dependency parsing with neural network features (Chen and Manning, 2014; Dyer et al., 2015; Cross and Huang, 2016; Kiperwasser and Goldberg, 2016; Dozat and Manning, 2016; Shi et al., 2017) has shown impressive performance. In particular, Kiperwasser and Goldberg (2016) used bidirectional LSTMs to generate features for individual words, which are then used to predict parsing actions. We base our model on Kiperwasser and Goldberg (2016) for both its simplicity and good performance.

Apart from dependency parsing, Abstract Meaning Representation (AMR) parsing (Flanigan et al., 2014; Werling et al., 2015; Wang et al., 2015; Konstas et al., 2017) may also benefit scene graph generation. However, as first pointed out in Anderson et al. (2016), the use of dependency trees still appears to be a common theme in the literature, and we leave the exploration of AMR parsing for scene graph generation as future work.

More broadly, our task also relates to entity and relation extraction, e.g. Katiyar and Cardie (2017), but there object attributes are not handled. Neural module networks (Andreas et al., 2016) also use dependency parses, but they translate questions into a series of actions, whereas we parse descriptions into their graph form. Finally, Krishnamurthy and Kollar (2013) connected parsing and grounding by training the parser in a weakly supervised fashion.

3 Task Description

In this section, we begin by reviewing the scene graph representation, and show how its nodes and edges relate to the words and arcs in dependency parsing. We then describe simple yet reliable rules to align nodes in scene graphs with words in textual descriptions, such that customized dependency parsing, described in the next section, may be trained and applied.

3.1 Scene Graph Definition

There are three types of nodes in a scene graph: object, attribute, and relation. Let $\mathcal{O}$ be the set of object classes, $\mathcal{A}$ be the set of attribute types, and $\mathcal{R}$ be the set of relation types. Given a sentence s, our goal in this paper is to parse s into a scene graph:

$$G(s) = \langle O(s), A(s), R(s) \rangle \tag{1}$$

where $O(s) = \{o_1(s), \ldots, o_m(s)\}, o_i(s) \in \mathcal{O}$ is the set of object instances mentioned in s, $A(s) \subseteq O(s) \times \mathcal{A}$ is the set of attributes associated with object instances, and $R(s) \subseteq O(s) \times \mathcal{R} \times O(s)$ is the set of relations between object instances.

$G(s)$ is a graph because we can first create an object node for every element in $O(s)$; then for every (o, a) pair in $A(s)$, we create an attribute node and add an unlabeled edge $o \rightarrow a$; finally for every (o_1, r, o_2) triplet in $R(s)$, we create a relation node and add two unlabeled edges $o_1 \rightarrow r$ and $r \rightarrow o_2$. The resulting directed graph exactly encodes information in $G(s)$. We call this the **node-centric** graph representation of a scene graph.

We realize that a scene graph can be equivalently represented by no longer distinguishing between the three types of nodes, yet assigning labels to the edges instead. Concretely, this means there is now only one type of node, but we assign a `ATTR` label for every $o \rightarrow a$ edge, a `SUBJ` label for every $o_1 \rightarrow r$ edge, and a `OBJT` label for every $r \rightarrow o_2$ edge. We call this the **edge-centric** graph representation of a scene graph.

We can now establish a connection between scene graphs and dependency trees. Here we only consider scene graphs that are acyclic[2]. The edge-centric view of a scene graph is very similar to a dependency tree: they are both directed acyclic graphs where the edges/arcs have labels. The difference is that in a scene graph, the nodes are the objects/attributes/relations and the edges have label space {`ATTR`, `SUBJ`, `OBJT`}, whereas in a dependency tree, the nodes are individual words in a sentence and the edges have a much larger label space.

3.2 Sentence-Graph Alignment

We have shown the connection between nodes in scene graphs and words in dependency parsing. With alignment between nodes in scene

[2]In Visual Genome, only 4.8% region graphs have cyclic structures.

graphs and words in the textual description, scene graph generation and dependency parsing becomes equivalent: we can construct the generated scene graph from the set of labeled edges returned by the dependency parser. Unfortunately, such alignment is not provided between the region graphs and region descriptions in the Visual Genome (Krishna et al., 2017) dataset. Here we describe how we use simple yet reliable rules to do sentence-graph (word-node) alignment.

There are two strategies that we could use in deciding whether to align a scene graph node d (whose label space is $\mathcal{O} \cup \mathcal{A} \cup \mathcal{R}$) with a word/phrase w in the sentence:

- Word-by-word match (WBW): $d \leftrightarrow w$ only when d's label and w match word-for-word.
- Synonym match (SYN)[3]: $d \leftrightarrow w$ when the wordnet synonyms of d's label contain w.

Obviously WBW is a more conservative strategy than SYN.

We propose to use two cycles and each cycle further consists of three steps, where we try to align objects, attributes, relations in that order. The pseudocode for the first cycle is in Algorithm 1. The second cycle repeats line 4-15 immediately afterwards, except that in line 6 we also allow SYN. Intuitively, in the first cycle we use a conservative strategy to find "safe" objects, and then scan for their attributes and relations. In the second cycle we relax and allow synonyms in aligning object nodes, also followed by the alignment of attribute and relation nodes.

The ablation study of the alignment procedure is reported in the experimental section.

4 Customized Dependency Parsing

In the previous section, we have established the connection between scene graph generation and dependency parsing, which assigns a parent word for every word in a sentence, as well as a label for this directed arc. We start by describing our base dependency parsing model, which is neural network based and performs among the state-of-the-art. We then show why and how we do customization, such that scene graph generation is achieved with a single, end-to-end model.

[3]This strategy is also used in (Denkowski and Lavie, 2014) and (Anderson et al., 2016).

Algorithm 1: First cycle of the alignment procedure.

1 **Input**: Sentence s; Scene graph $G(s)$
2 Initialize aligned nodes N as empty set
3 Initialize aligned words W as empty set
4 **for** o in object nodes of $G(s) \setminus N$ **do**
5 **for** w in $s \setminus W$ **do**
6 **if** $o \leftrightarrow w$ according to WBW **then**
7 Add (o, w); $N = N \cup \{o\}$; $W = W \cup \{w\}$
8 **for** a in attribute nodes of $G(s) \setminus N$ **do**
9 **for** w in $s \setminus W$ **do**
10 **if** $a \leftrightarrow w$ according to WBW or SYN **and** a's object node is in N **then**
11 Add (a, w); $N = N \cup \{a\}$; $W = W \cup \{w\}$
12 **for** r in relation nodes of $G(s) \setminus N$ **do**
13 **for** w in $s \setminus W$ **do**
14 **if** $r \leftrightarrow w$ according to WBW or SYN **and** r's subject and object nodes are both in N **then**
15 Add (r, w); $N = N \cup \{r\}$; $W = W \cup \{w\}$

4.1 Neural Dependency Parsing Base Model

We base our model on the transition-based parser of Kiperwasser and Goldberg (2016). Here we describe its key components: the arc-hybrid system that defines the transition actions, the neural architecture for feature extractor and scoring function, and the loss function.

The Arc-Hybrid System In the arc-hybrid system, a configuration consists of a stack σ, a buffer β, and a set T of dependency arcs. Given a sentence $s = w_1, \ldots, w_n$, the system is initialized with an empty stack σ, an empty arc set T, and $\beta = 1, \ldots, n, \texttt{ROOT}$, where ROOT is a special index. The system terminates when σ is empty and β contains only ROOT. The dependency tree is given by the arc set T upon termination.

The arc-hybrid system allows three transition actions, SHIFT, LEFT$_l$, RIGHT$_l$, described in Table 1. The SHIFT transition moves the first element of the buffer to the stack. The LEFT(l) transition yields an arc from the first element of the buffer to the top element of the stack, and then removes the top element from the stack. The

Stack σ_t	Buffer β_t	Arc set T_t	Action	Stack σ_{t+1}	Buffer β_{t+1}	Arc set T_{t+1}
σ	$b_0\|\beta$	T	SHIFT	$\sigma\|b_0$	β	T
$\sigma\|s_1\|s_0$	$b_0\|\beta$	T	LEFT(l)	$\sigma\|s_1$	$b_0\|\beta$	$T \cup \{(b_0, s_0, l)\}$
$\sigma\|s_1\|s_0$	β	T	RIGHT(l)	$\sigma\|s_1$	β	$T \cup \{(s_1, s_0, l)\}$
$\sigma\|s_0$	β	T	REDUCE	σ	β	T

Table 1: Transition actions under the arc-hybrid system. The first three actions are from dependency parsing; the last one is introduced for scene graph parsing.

RIGHT(l) transition yields an arc from the second top element of the stack to the top element of the stack, and then also removes the top element from the stack.

The following paragraphs describe how to select the correct transition action (and label l) in each step in order to generate a correct dependency tree.

BiLSTM Feature Extractor Let the word embeddings of a sentence s be $\mathbf{w}_1, \ldots, \mathbf{w}_n$. An LSTM cell is a parameterized function that takes as input $\mathbf{w}_t$, and updates its hidden states:

$$\text{LSTM cell} : (\mathbf{w}_t, \mathbf{h}_{t-1}) \rightarrow \mathbf{h}_t \quad (2)$$

As a result, an LSTM network, which simply applies the LSTM cell t times, is a parameterized function mapping a sequence of input vectors $\mathbf{w}_{1:t}$ to a sequence of output vectors $\mathbf{h}_{1:t}$. In our notation, we drop the intermediate vectors $\mathbf{h}_{1:t-1}$ and let $\text{LSTM}(\mathbf{w}_{1:t})$ represent $\mathbf{h}_t$.

A bidirectional LSTM, or BiLSTM for short, consists of two LSTMs: LSTM_F which reads the input sequence in the original order, and LSTM_B which reads it in reverse. Then

$$\text{BILSTM}(\mathbf{w}_{1:n}, i) = \text{LSTM}_F(\mathbf{w}_{1:i}) \circ \text{LSTM}_B(\mathbf{w}_{n:i}) \quad (3)$$

where $\circ$ denotes concatenation. Intuitively, the forward LSTM encodes information from the left side of the i-th word and the backward LSTM encodes information to its right, such that the vector $\mathbf{v}_i = \text{BILSTM}(\mathbf{w}_{1:n}, i)$ has the full sentence as context.

When predicting the transition action, the feature function $\phi(c)$ that summarizes the current configuration $c = (\sigma, \beta, T)$ is simply the concatenated BiLSTM vectors of the top three elements in the stack and the first element in the buffer:

$$\phi(c) = \mathbf{v}_{s_2} \circ \mathbf{v}_{s_1} \circ \mathbf{v}_{s_0} \circ \mathbf{v}_{b_0} \quad (4)$$

MLP Scoring Function The score of transition action y under the current configuration c is determined by a multi-layer perceptron with one hidden layer:

$$f(c, y) = MLP(\phi(c))[y] \quad (5)$$

where

$$MLP(x) = W_2 \cdot \tanh(W_1 \cdot x + b_1) + b_2 \quad (6)$$

Hinge Loss Function The training objective is to raise the scores of correct transitions above scores of incorrect ones. Therefore, at each step, we use a hinge loss defined as:

$$\mathcal{L} = \max(0, 1 - \max_{y^+ \in Y^+} f(c, y^+) + \max_{y^- \in Y \setminus Y^+} f(c, y^-)) \quad (7)$$

where Y is the set of possible transitions and Y^+ is the set of correct transitions at the current step. In each training step, the parser scores all possible transitions using Eqn. 5, incurs a loss using Eqn. 7, selects a following transition, and updates the configuration. Losses at individual steps are summed throughout the parsing of a sentence, and then parameters are updated using backpropagation.

In test time, we simply choose the transition action that yields the highest score at each step.

4.2 Customization

In order to generate scene graphs with dependency parsing, modification is necessary for at least two reasons. First, we need to redefine the label space of arcs so as to reflect the edge-centric representation of a scene graph. Second, not every word in the sentence will be (part of) a node in the scene graph (see Figure 2 for an example). In other words, some words in the sentence may not have a parent word, which violates the dependency parsing setting. We tackle these two challenges by redesigning the edge labels and expanding the set of transition actions.

Redesigning Edge Labels We define a total of five edge labels, so as to faithfully bridge the edge-centric view of scene graphs with dependency parsing models:

- CONT: This label is created for nodes whose label is a phrase. For example, the phrase "in front of" is a single relation node in the scene graph. By introducing the CONT label, we expect the parsing result to be either

$$\text{in} \xrightarrow{\text{CONT}} \text{front} \xrightarrow{\text{CONT}} \text{of} \tag{8}$$

  or

$$\text{in} \xleftarrow{\text{CONT}} \text{front} \xleftarrow{\text{CONT}} \text{of} \tag{9}$$

  where the direction of the arcs (left or right) is predefined by hand.

  The leftmost word under the right arc rule or the rightmost word under the left arc rule is called the *head* of the phrase. A single-word node does not need this CONT label, and the head is itself.

- ATTR: The arc label from the head of an object node to the head of an attribute node.

- SUBJ: The arc label from the head of an object node (subject) to the head of a relation node.

- OBJT: The arc label from the head of a relation node to the head of an object node (object).

- BEGN: The arc label from the ROOT index to all heads of object nodes without a parent.

Expanding Transition Actions With the three transition actions SHIFT, LEFT(l), RIGHT(l), we only drop an element (from the top of the stack) after it has already been associated with an arc. This design ensures that an arc is associated with every word. However, in our setting for scene graph generation, there may be no arc for some of the words, especially empty words.

Our solution is to augment the action set with a REDUCE action, that pops the stack *without* adding to the arc set (see Table 1). This action is often used in other transition-based dependency parsing systems (e.g. arc-eager (Nivre, 2004)). More recently, Hershcovich et al. (2017) and Buys and Blunsom (2017) also included this action when parsing sentences to graph structures.

Parser	F-score
Stanford (Schuster et al., 2015)	0.3549
SPICE (Anderson et al., 2016)	0.4469
Ours (left arc rule)	**0.4967**
Ours (right arc rule)	0.4952
Ours (all SYN)	0.4877
Ours (no SYN)	0.4538
Oracle	0.6985

Table 2: The F-scores (i.e. SPICE metric) between scene graphs parsed from region descriptions and ground truth region graphs on the intersection of Visual Genome (Krishna et al., 2017) and MS COCO (Lin et al., 2014) validation set.

We still minimize the loss function defined in Eqn. 7, except that now $|Y|$ increases from 3 to 4. During training, we impose the oracle to select the REDUCE action when it is in Y^+. In terms of loss function, we increment by 1 the loss incurred by the other 3 transition actions if REDUCE incurs zero loss.

5 Experiments

5.1 Implementation Details

We train and evaluate our scene graph parsing model on (a subset of) the Visual Genome (Krishna et al., 2017) dataset. Each image in Visual Genome contains a number of regions, and each region is annotated with both a region description and a region scene graph. Our training set is the intersection of Visual Genome and MS COCO (Lin et al., 2014) train2014 set, which contains a total of 34027 images/ 1070145 regions. We evaluate on the intersection of Visual Genome and MS COCO val2014 set, which contains a total of 17471 images/ 547795 regions.

In our experiments, the number of hidden units in BiLSTM is 256; the number of layers in BiLSTM is 2; the word embedding dimension is 200; the number of hidden units in MLP is 100. We use fixed learning rate 0.001 and Adam optimizer (Kingma and Ba, 2014) with epsilon 0.01. Training usually converges within 4 epochs.

We will release our code and trained model upon acceptance.

5.2 Quality of Parsed Scene Graphs

We use a slightly modified version of SPICE score (Anderson et al., 2016) to evaluate the quality of

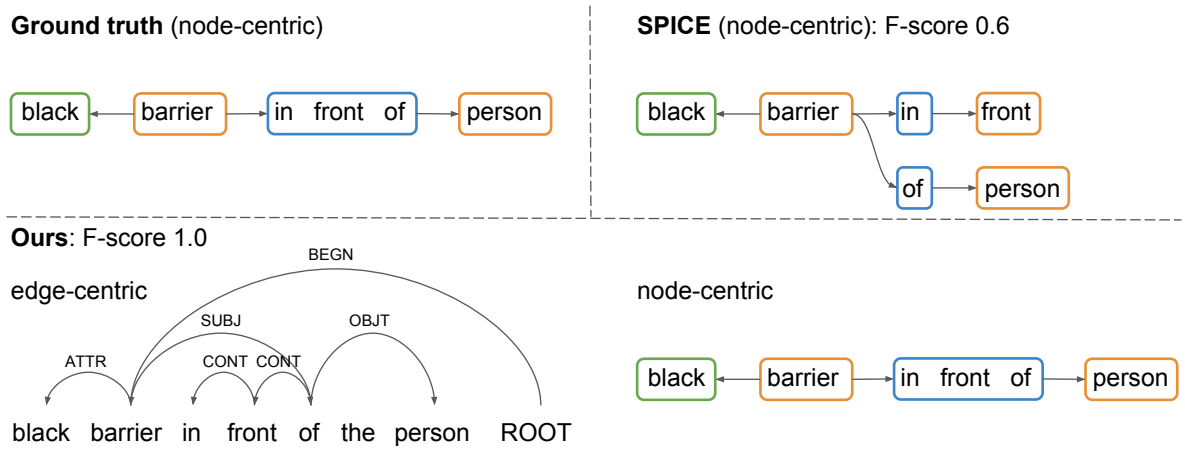

Figure 2: Scene graph parsing result of the sentence "black barrier in front of the person". In the node-centric graphs, orange represents object node, green represents attribute node, blue represents relation node.

	Stack	Buffer	Action
0		black barrier in front of the person ROOT	SHIFT
1	black	barrier in front of the person ROOT	LEFT(ATTR)
2		barrier in front of the person ROOT	SHIFT
3	barrier	in front of the person ROOT	SHIFT
4	barrier in	front of the person ROOT	LEFT(CONT)
5	barrier	front of the person ROOT	SHIFT
6	barrier front	of the person ROOT	LEFT(CONT)
7	barrier	of the person ROOT	SHIFT
8	barrier of	the person ROOT	SHIFT
9	barrier of the	person ROOT	REDUCE
10	barrier of	person ROOT	SHIFT
11	barrier of person	ROOT	RIGHT(OBJT)
12	barrier of	ROOT	RIGHT(SUBJ)
13	barrier	ROOT	LEFT(BEGN)
14		ROOT	

Figure 3: Intermediate actions taken by the trained dependency parser when parsing the sentence "black barrier in front of the person".

scene graph parsing. Specifically, for every region, we parse its description using a parser (e.g. the one used in SPICE or our customized dependency parser), and then calculate the F-score between the parsed graph and the ground truth region graph (see Section 3.2 of Anderson et al. (2016) for more details). Note that when SPICE calculates the F-score, a node in one graph could be matched to several nodes in the other, which is problematic. We fix this and enforce one-to-one matching when calculating the F-score. Finally, we report the average F-score across all regions.

Table 2 summarizes our results. We see that our customized dependency parsing model achieves an average F-score of 49.67%, which significantly outperforms the parser used in SPICE by 5 percent. This result shows that our customized dependency parser is very effective at learning from data, and generates more accurate scene graphs than the best previous approach.

Ablation Studies First, we study how the sentence-graph alignment procedure affects the final performance. Recall that our procedure involves two cycles, each with three steps. Of the six steps, synonym match (SYN) is only not used in the first step. We tried two more settings, where SYN is either used in all six steps or none of the six steps. We can see from Table 2 that the final

	Development set			Test set		
	R@5	**R@10**	**Med. rank**	**R@5**	**R@10**	**Med. rank**
(Schuster et al., 2015)	33.82%	45.58%	6	34.96%	45.68%	**5**
Ours	**36.69%**	**49.41%**	**4**	**36.70%**	**49.37%**	**5**

Table 3: Image retrieval results. We follow the same experiment setup as Schuster et al. (2015), except using a different scoring function when ranking images. Our parser consistently outperforms the Stanford Scene Graph Parser across evaluation metrics.

F-score drops in both cases, hence supporting the procedure that we chose.

Second, we study whether changing the direction of CONT arcs from pointing left to pointing right will make much difference. Table 2 shows that the two choices give very similar performance, suggesting that our dependency parser is robust to this design choice.

Finally, we report the oracle score, which is the similarity between the aligned graphs that we use during training and the ground truth graphs. The F-score is relatively high at 69.85%. This shows that improving the parser (about 20% margin) and improving the sentence-graph alignment (about 30% margin) are both promising directions for future research.

Qualitative Examples We provide one parsing example in Figure 2 and Figure 3. This is a sentence that is relatively simple, and the underlying scene graph includes two object nodes, one attribute node, and one compound word relation node. In parsing this sentence, all four actions listed in Table 1 are used (see Figure 3) to produce the edge-centric scene graph (bottom left of Figure 2), which is then trivially converted to the node-centric scene graph (bottom right of Figure 2).

5.3 Application in Image Retrieval

We test if the advantage of our parser can be propagated to computer vision tasks, such as image retrieval. We directly compare our parser with the Stanford Scene Graph Parser (Schuster et al., 2015) on the development set and test set of the image retrieval dataset used in Schuster et al. (2015) (not Visual Genome).

For every region in an image, there is a human-annotated region description and region scene graph. The queries are the region descriptions. If the region graph corresponding to the query is a subgraph of the complete graph of another image, then that image is added to the ground truth set for this query. All these are strictly following Schuster et al. (2015). However, since we did not obtain nor reproduce the CRF model used in Johnson et al. (2015) and Schuster et al. (2015), we used F-score similarity instead of the likelihood of the maximum a posteriori CRF solution when ranking the images based on the region descriptions. Therefore the numbers we report in Table 3 are not directly comparable with those reported in Schuster et al. (2015).

Our parser delivers better retrieval performance across all three evaluation metrics: recall@5, recall@10, and median rank. We also notice that the numbers in our retrieval setting are higher than those (even with oracle) in Schuster et al. (2015)'s retrieval setting. This strongly suggests that generating accurate scene graphs from images is a very promising research direction in image retrieval, and grounding parsed scene graphs to bounding box proposals without considering visual attributes/relationships (Johnson et al., 2015) is suboptimal.

6 Conclusion

In this paper, we offer a new perspective and solution to the task of parsing scene graphs from textual descriptions. We begin by moving the labels/types from the nodes to the edges and introducing the edge-centric view of scene graphs. We further show that the gap between edge-centric scene graphs and dependency parses can be filled with a careful redesign of label and action space. This motivates us to train a single, customized, end-to-end neural dependency parser for this task, as opposed to prior approaches that used generic dependency parsing followed by heuristics or simple classifier. We directly train our parser on a subset of Visual Genome (Krishna et al., 2017), without transferring any knowledge from Penn Treebank (Marcus et al., 1993) as previous works did.

The quality of our trained parser is validated in terms of both SPICE similarity to the ground truth graphs and recall rate/median rank when performing image retrieval.

We hope our paper can lead to more thoughts on the creative uses and extensions of existing NLP tools to tasks and datasets in other domains. In the future, we plan to tackle more computer vision tasks with this improved scene graph parsing technique in hand, such as image region grounding. We also plan to investigate parsing scene graph with cyclic structures, as well as whether/how the image information can help boost parsing quality.

Acknowledgments

The majority of this work was done when YSW and XZ were visiting Johns Hopkins University. We thank Peter Anderson, Sebastian Schuster, Ranjay Krishna, Tsung-Yi Lin for comments and help regarding the experiments. We also thank Tianze Shi, Dingquan Wang, Chu-Cheng Lin for discussion and feedback on the draft. This work was sponsored by the National Science Foundation Center for Brains, Minds, and Machines NSF CCF-1231216. CL also acknowledges an award from Snap Inc.

References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. SPICE: semantic propositional image caption evaluation. In *ECCV*. Springer, volume 9909 of *Lecture Notes in Computer Science*, pages 382–398.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *CVPR*. IEEE Computer Society, pages 39–48.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: visual question answering. In *ICCV*. IEEE Computer Society, pages 2425–2433.

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *ACL*. Association for Computational Linguistics, pages 1215–1226.

Angel X. Chang, Manolis Savva, and Christopher D. Manning. 2014. Learning spatial knowledge for text to 3d scene generation. In *EMNLP*. ACL, pages 2028–2038.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*. ACL, pages 740–750.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*. ACL, pages 1724–1734.

James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional LSTM. In *ACL (2)*. The Association for Computer Linguistics.

Bo Dai, Yuqi Zhang, and Dahua Lin. 2017. Detecting visual relationships with deep relational networks. In *CVPR*. IEEE Computer Society, pages 3298–3308.

Michael J. Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *WMT@ACL*. The Association for Computer Linguistics, pages 376–380.

Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Trevor Darrell, and Kate Saenko. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*. IEEE Computer Society, pages 2625–2634.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*. JMLR.org, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 647–655.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR* abs/1611.01734.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *ACL (1)*. The Association for Computer Linguistics, pages 334–343.

Jeffrey Flanigan, Sam Thomson, Jaime G. Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *ACL (1)*. The Association for Computer Linguistics, pages 1426–1436.

Andrea Frome, Gregory S. Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. 2013. Devise: A deep visual-semantic embedding model. In *NIPS*. pages 2121–2129.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *ACL*. Association for Computational Linguistics, pages 1127–1138.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. 2016. Natural language object retrieval. In *CVPR*. IEEE Computer Society, pages 4555–4564.

Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Fei-Fei Li. 2015. Image retrieval using scene graphs. In *CVPR*. IEEE Computer Society, pages 3668–3678.

Lukasz Kaiser, Aidan N. Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. 2017. One model to learn them all. *CoRR* abs/1706.05137.

Andrej Karpathy and Fei-Fei Li. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*. IEEE Computer Society, pages 3128–3137.

Arzoo Katiyar and Claire Cardie. 2017. Going out on a limb: Joint extraction of entity mentions and relations without dependency trees. In *ACL*. Association for Computational Linguistics, pages 917–928.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL* 4:313–327.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: sequence-to-sequence models for parsing and generation. In *ACL (1)*. Association for Computational Linguistics, pages 146–157.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision* 123(1):32–73.

Jayant Krishnamurthy and Thomas Kollar. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *TACL* 1:193–206.

Sandra Kübler, Ryan T. McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Yikang Li, Wanli Ouyang, Bolei Zhou, Kun Wang, and Xiaogang Wang. 2017. Scene graph generation from objects, phrases and caption regions. *CoRR* abs/1707.09700.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: common objects in context. In *ECCV*. Springer, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755.

Chenxi Liu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, and Alan L. Yuille. 2017a. Recurrent multimodal interaction for referring image segmentation. In *ICCV*. IEEE Computer Society, pages 1280–1289.

Chenxi Liu, Junhua Mao, Fei Sha, and Alan L. Yuille. 2017b. Attention correctness in neural image captioning. In *AAAI*. AAAI Press, pages 4176–4182.

Cewu Lu, Ranjay Krishna, Michael S. Bernstein, and Fei-Fei Li. 2016. Visual relationship detection with language priors. In *ECCV*. Springer, volume 9905 of *Lecture Notes in Computer Science*, pages 852–869.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L. Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In *CVPR*. IEEE Computer Society, pages 11–20.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan L. Yuille. 2014. Deep captioning with multimodal recurrent neural networks (m-rnn). *CoRR* abs/1412.6632.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics* 19(2):313–330.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*. Association for Computational Linguistics, pages 50–57.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. ACL, pages 1532–1543.

Sebastian Schuster, Ranjay Krishna, Angel Chang, Li Fei-Fei, and Christopher D Manning. 2015. Generating semantically precise scene graphs from textual descriptions for improved image retrieval. In *Proceedings of the fourth workshop on vision and language*. volume 2.

Tianze Shi, Liang Huang, and Lillian Lee. 2017. Fast(er) exact decoding and global training for transition-based dependency parsing via a minimal feature set. In *EMNLP*. Association for Computational Linguistics, pages 12–23.

Damien Teney, Lingqiao Liu, and Anton van den Hengel. 2016. Graph-structured representations for visual question answering. *CoRR* abs/1609.05600.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2015. Order-embeddings of images and language. *CoRR* abs/1511.06361.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. A transition-based algorithm for AMR parsing. In *HLT-NAACL*. The Association for Computational Linguistics, pages 366–375.

Keenon Werling, Gabor Angeli, and Christopher D. Manning. 2015. Robust subgraph generation improves abstract meaning representation parsing. In *ACL (1)*. The Association for Computer Linguistics, pages 982–991.

Danfei Xu, Yuke Zhu, Christopher B. Choy, and Li Fei-Fei. 2017. Scene graph generation by iterative message passing. In *CVPR*. IEEE Computer Society, pages 3097–3106.

Yuke Zhu, Oliver Groth, Michael S. Bernstein, and Li Fei-Fei. 2016. Visual7w: Grounded question answering in images. In *CVPR*. IEEE Computer Society, pages 4995–5004.