

Explicitly Representing Syntax Improves Sentence-to-Layout Prediction of Unexpected Situations

Wolf Nuyts* Ruben Cartuyvels*,† Marie-Francine Moens

Department of Computer Science

KU Leuven, Belgium

wolf.nuyts@gmail.com, {ruben.cartuyvels, sien.moens}@kuleuven.be

Abstract

Recognizing visual entities in a natural language sentence and arranging them in a 2D spatial layout require a compositional understanding of language and space. This task of layout prediction is valuable in text-to-image synthesis as it allows localized and controlled in-painting of the image. In this comparative study it is shown that we can predict layouts from language representations that implicitly or explicitly encode sentence syntax, if the sentences mention similar entity-relationships to the ones seen during training. To test compositional understanding, we collect a test set of grammatically correct sentences and layouts describing compositions of entities and relations that unlikely have been seen during training. Performance on this test set substantially drops, showing that current models rely on correlations in the training data and have difficulties in understanding the structure of the input sentences. We propose a novel structural loss function that better enforces the syntactic structure of the input sentence and show large performance gains in the task of 2D spatial layout prediction conditioned on text. The loss has the potential to be used in other generation tasks where a tree-like structure underlies the conditioning modality. Code, trained models, and the USCOCO evaluation set are available via Github.¹

1 Introduction

Current neural networks and especially transformer architectures pretrained on large amounts of data build powerful representations of content. However, unlike humans, they fail when confronted with unexpected situations and content which is out of context (Geirhos et al., 2020). Compositionality is considered a powerful tool in

human cognition as it enables humans to understand and generate a potentially infinite number of novel situations by viewing the situation as a novel composition of familiar simpler parts (Humboldt, 1999; Chomsky, 1965; Frankland and Greene, 2020).

In this paper we hypothesize that representations that better encode the syntactical structure of a sentence—in our case a constituency tree—are less sensitive to a decline in performance when confronted with unexpected situations. We test this hypothesis with the task of 2D visual object layout prediction given a natural language input sentence (Figure 2 gives an overview of the task and of our models). We collect a test set of grammatically correct sentences and layouts (visual “imagined” situations), called Unexpected Situations of Common Objects in Context (USCOCO) describing compositions of entities and relations that unlikely have been seen during training. Most importantly, we propose a novel structural loss function that better retains the syntactic structure of the sentence in its representation by enforcing the alignment between the syntax tree embeddings and the output of the downstream task, in our case the visual embeddings. This loss function is evaluated both with models that explicitly integrate syntax (i.e., linearized constituent trees using brackets and tags as tokens) and with models that implicitly encode syntax (i.e., language models trained with a transformer architecture). Models that explicitly integrate syntax show large performance gains in the task of 2D spatial layout prediction conditioned on text when using the proposed structural loss.

The task of layout prediction is valuable in text-to-image synthesis as it allows localized and controlled in-painting of the image. Apart from measuring the understanding of natural language by the machine (Ulinski, 2019), text-to-image synthesis is popular because of the large application

*Joint first authors

†Corresponding author.

¹<https://github.com/rubencart/USCOCO>.

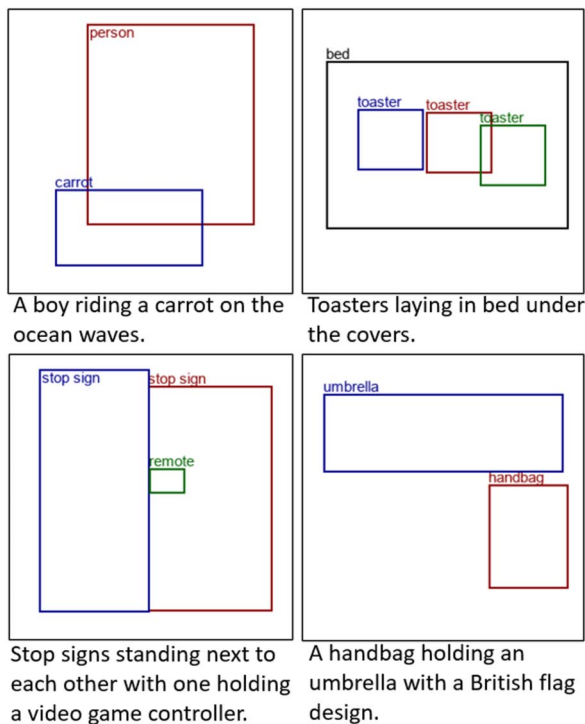


Figure 1: Samples from the USCOCO dataset.

potential (e.g., when creating games and virtual worlds). Current generative diffusion models create a naturalistic image from a description in natural language (e.g., DALL-E 2, Ramesh et al., 2022), but lack local control in the image triggered by the role a word has in the interpretation of the sentence (Rassin et al., 2022), and fail to adhere to specified relations between objects (Gokhale et al., 2022). Additionally, if you change the description, for instance, by changing an object name or its attribute, a new image is generated from scratch, instead of a locally changed version of the current image, which is a concern in research (e.g., Couairon et al., 2022; Poole et al., 2022). Chen et al. (2023) and Qu et al. (2023) show that using scene layouts as additional input greatly improves the spatial reasoning of text-to-image models, substantiating the importance of the text-to-layout synthesis task. We restrict the visual scene to the spatial 2D arrangements of objects mentioned in the natural language sentence, taking into account the size and positions of the objects (Figure 1). From these layouts images can be generated that accurately adhere to the spatial restrictions encoded by the layouts (Chen et al., 2023; Qu et al., 2023), but this is not within the scope of this paper. We emphasize that while we argue that good layout predictions are valuable, the question this

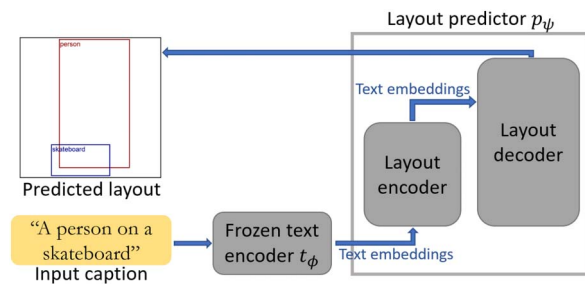


Figure 2: Overview of text-to-layout prediction.

study aims to answer is not how to build the best possible layout predictors, but whether and if so how explicitly representing syntax can improve such predictors, especially with respect to their robustness to unseen and unexpected inputs.

The contributions of our research are the following: (1) We introduce a new test set called USCOCO for the task of 2D visual layout prediction based on a natural language sentence containing unusual interactions between known objects; (2) We compare multiple sentence encoding models that implicitly and explicitly integrate syntactical structure in the neural encoding, and evaluate them with the downstream task of layout prediction; (3) We introduce a novel parallel decoder for layout prediction based on transformers; (4) We propose a novel contrastive structural loss that enforces the encoding of syntax in the representation of a visual scene and show that it increases generalization to unexpected compositions; and (5) We conduct a comprehensive set of experiments, using quantitative metrics, human evaluation and probing to evaluate the encoding of structure, more specifically syntax.

2 Related Work

Implicit Syntax in Language and

Visio-linguistic Models Deep learning has ruled out the need for feature engineering in natural language processing including the extraction of syntactical features. With the advent of contextualized language models pretrained on large text corpora (Peters et al., 2018; Devlin et al., 2019), representations of words and sentences are dynamically computed. Several studies have evaluated syntactical knowledge embedded in language models through targeted syntactic evaluation, probing, and downstream natural language understanding tasks (Hewitt and Manning, 2019; Manning et al., 2020; Linzen and Baroni,

2021; Kulmizev and Nivre, 2021). Here we consider the task of visually imagining the situation expressed in a sentence and show that in case of unexpected situations that are unlikely to occur in the training data the performance of language models strongly decreases.

Probing Compositional Multimodal Reasoning

One of the findings of our work, that is, the failure of current models in sentence-to-layout prediction of unusual situations, are in line with recent text-image alignment test suites like Winoground (Thrush et al., 2022) and VALSE (Parcalabescu et al., 2022), that ask to correctly retrieve an image using grammatically varying captions. The benchmark of Gokhale et al. (2022) considers a generation setting, like us, however, they evaluate image generation (down to pixels), while ignoring the role of the language encoder.² Moreover, their captions are automatically generated from object names and simple spatial relations, and hence only contain explicit spatial relations, whereas our USCOCO captions are modified human-written COCO captions and include implicit spatial relations.

Explicitly Embedding Syntax in Language Representations

Popa et al. (2021) use a tensor factorisation model for computing token embeddings that integrate dependency structures. In section 3.2.1 we discuss models that integrate a constituency parse tree (Qian et al., 2021; Sartran et al., 2022) as we use them as encoders in the sentence-to-layout prediction task. They build on generative parsing approaches using recurrent neural networks (Dyer et al., 2016; Choe and Charniak, 2016).

Visual Scene Layout Prediction Hong et al. (2018), Tan et al. (2018), and Li et al. (2019b) introduce layout predictors that are similar to our autoregressive model. They also train on the realistic COCO dataset and generate a dynamic number of objects, but they do not investigate the layout predictions. Other layout prediction methods require structured input like triplets or graphs instead of free-form text (Li et al., 2019a; Johnson et al., 2018; Lee et al., 2020), are confined to layouts of only 2 objects (Collell et al., 2021), are unconditional (Li et al., 2019b), work in simplified, non-realistic settings (Radevski et al., 2020;

²Their data has not been made publicly available at the time of writing.

Lee et al., 2020), or focus on predicting positions for known objects and their relationships (Radevski et al., 2020).

3 Methods

3.1 Task Definition

Given a natural language caption C , the task is to generate a layout L that captures a spatial 2D visual arrangement of the objects that the caption describes. A layout $L = \{b_i\}_i = \{(o_i, x_i, y_i, w_i, h_i)\}_i$ consists of a varying number of visual objects, each represented by a 5-tuple, where o_i is a label from a category vocabulary (in this paper: one of the 80 COCO categories, e.g., “elephant”), and where (x_i, y_i, w_i, h_i) refers to the bounding box for that object. The coordinates of the middle point of a box are x_i, y_i , and w_i, h_i are the width and height. A caption C consists of a number N_c of word tokens c_i . Hence, we want to learn the parameters θ of a model f_θ that maps captions to layouts: $L = f_\theta(C)$.

Model Overview We split the prediction problem into two parts. First, a text encoder t_ϕ computes (potentially structured) text embeddings e_j for input word tokens c_i : $E = t_\phi(C)$. Second, a layout predictor p_ψ predicts an embedding v_k per visual object: $v_k = p_\psi(E)$. These are projected by a multilayer perceptron to a categorical probability distribution to predict object labels: $o_k \sim \text{softmax}(\text{MLP}^{\text{label}}(v_k))$. Regression ($\times 4$) is used for positions $b_k = (x_k, y_k, w_k, h_k)$: e.g., $x_k = \sigma(\text{MLP}^{\text{pos}}(v_k)_x)$, with $\sigma : \mathbb{R}^d \mapsto [0, 1]$ a sigmoid.

3.2 Text Encoders t_ϕ

We consider two types of text encoders. First, we explicitly encode the syntactical structure of a sentence in its semantic representation. As syntactical structure we choose a constituency parse as it naturally represents the structure of human language, and phrases can be mapped to visual objects. Human language is characterized by recursive structures which correspond with the recursion that humans perceive in the world (Hawkings, 2021; Hauser et al., 2002). Second, we encode the sentence with a state-of-the-art sentence encoder that is pretrained with a next token prediction objective. We assume that it implicitly encodes syntax (Tenney et al., 2019; Warstadt et al., 2020). The choice of the next-token prediction objective is motivated by existing work

that compares language models with implicit vs. explicit syntax (Qian et al., 2021; Sartran et al., 2022). We use pretrained text encoders, which are all frozen during layout prediction training to allow a clean comparison.

3.2.1 Explicitly Embedding Syntax in Sentence Representations

The models that explicitly embed syntax take a linearized version of the constituency trees as input, using brackets and constituent tags as tokens in addition to the input sentence C , to end up with C^{lin} . For instance, “*A dog catches a frisbee*” would be preprocessed into “(NP *a dog*) (VP *catches (NP a frisbee)*)”. The constituency trees are obtained with the parser of Kitaev and Klein (2018) and Kitaev et al. (2019). The model computes an embedding e_j^{struct} ($0 \leq j < N_c + N_{\text{lin}}$) per input token c_j (including the parentheses and syntax tags). The embeddings are given as is to the layout predictor, and since tokens explicitly representing structure (parentheses, syntax tags) have their own embedding, we assume the sequence of e_j^{struct} to carry more structural information than the sequence formed by e_i^{base} (cfr. section 3.2.2). We consider the following models, which are all pretrained on the BLLIP_{LG} dataset ($\approx 40\text{M}$ tokens Charniak et al., 2000).

PLM: The Parsing as Language Model from Qian et al. (2021) inputs C^{lin} into an untrained GPT-2 model and learns e_j^{struct} by training on a next token prediction task.

PLM_{mask}: This model, of Qian et al. (2021), is similar to PLM but uses masking to constrain two of the attention heads in the transformer layers to attend to tokens that are respectively part of the current constituent and part of the rest of the partially parsed sentence.

TG: The Transformer Grammars model from Sartran et al. (2022) uses a masking scheme that constrains all attention heads to only attend to local parts of the constituent tree. This results in a recursive composition of the representations of smaller constituents into the representations of larger constituents, which reflects closely the recursive property of Recurrent Neural Network Grammars (RNNG) models (Dyer et al., 2016). We adapt this model to use a GPT-2 backbone and we train it for next token prediction on the same dataset as the PLM models for a fair comparison.

TG_{RB}: To test to what extent differences in layout prediction performance are due to the explicit use of a constituency grammar, and not a byproduct of model and/or input differences, this model uses trivial right-branching constituency trees (constructed by taking the silver trees and moving all closing brackets to the end of the sentence), instead of silver constituency trees, both during pretraining and during layout generation. This model is also used by Sartran et al. (2022) as ablation baseline.

3.2.2 Baselines That Are Assumed to Implicitly Encode Syntax

The baselines take a sequence of text tokens c_i ($0 \leq i < N_c$) and produce a sequence of the same length, of embeddings e_i^{base} ($0 \leq i < N_c$), which will be given to the layout decoder.

GPT-2_{Bllip}: This language model is also used by Qian et al. (2021) and shares its architecture and training regime with GPT-2 (Radford et al., 2019). It is trained on the sentences (not the linearized parse trees) of the BLLIP_{LG} dataset to predict the next token given the history (Charniak et al., 2000). Hence, this model is trained on the same sentences as the models of section 3.2.1. Even though it is debatable whether transformers can learn implicit syntax from the relatively small ($\approx 40\text{M}$ tokens) BLLIP_{LG} dataset, this model is used as baseline by existing work on explicit syntax in language modeling (Qian et al., 2021; Sartran et al., 2022), which is why we also include it. Furthermore, there is evidence that pretraining datasets of 10M-100M tokens suffice for transformers to learn most of their syntax capabilities (Pérez-Mayos et al., 2021; Zhang et al., 2021; Samuel et al., 2023), even though orders of magnitude ($> 1\text{B}$) more pretraining tokens are required for more general downstream NLU tasks.

GPT-2: As published by Radford et al. (2019), trained for next token prediction on a large-scale scraped webtext dataset.

GPT-2_{Bllip}^{shuffle}: Identical to the GPT-2_{Bllip} model but the tokens in the input sentence are randomly shuffled, to test whether syntax has any contribution at all. The pretraining is exactly the same as GPT-2_{Bllip}, with the token order preserved.

LLaMA: Large state-of-the-art language models trained on massive amounts of text, we use the 7B and 33B model variants (Touvron et al., 2023).

3.3 Layout Predictors p_ψ

3.3.1 Models

As a baseline we consider the layout prediction LSTM from Hong et al. (2018) and Li et al. (2019c), further referred to as ObjLSTM.³ This model does not perform well and it trains slowly because of the LSTM architecture, so we propose two novel layout predictors. The two models use the same transformer encoder, and differ in their transformer decoder architecture.

PAR: This decoding model is inspired by the DETR model for object detection of Carion et al. (2020). The decoder generates a number of objects in a single forward parallel pass, after first predicting the number of objects.

SEQ: This autoregressive model is similar to the language generating transformer of Vaswani et al. (2017), but decodes object labels and bounding boxes and not language tokens. It predicts an object per step until the end-of-sequence token is predicted, or the maximum length is reached. The model is similar to the layout decoder of Li et al. (2019c), but uses transformers instead of LSTMs.

3.3.2 Training

The PAR Model is trained analogous to Carion et al. (2020) and Stewart et al. (2016) by first computing the minimum cost bipartite matching between predicted objects \hat{b}_i and ground-truth objects b_j (as the ordering might differ), using the Hungarian algorithm (Kuhn, 2010), with differences in box labels, positions, and overlaps as cost.

The SEQ Model is trained to predict the next visual object given the previous GT objects. The order of generation is imposed (by a heuristic: from large to small in area, after Li et al. [2019c]). The 1st, 2nd, . . . generated object is matched with the largest, 2nd but largest, . . . ground-truth object.

The PAR and SEQ models apply the following losses to each matched pair of predicted box \hat{b}_i and ground-truth box b_j .

A Cross-entropy Loss $\mathcal{L}_{\text{label}}(\hat{b}_i, b_j)$ applied to the object labels.

A Combination of Regression Losses applied to the bounding box coordinates.

³This is the only existing model we found that generates varying numbers of bounding boxes from free-form text for the COCO dataset.

- An L1-loss $\mathcal{L}_{L1}(\hat{b}_i, b_j)$ applied to each of the dimensions of the boxes (Carion et al., 2020).
- The generalized box IoU loss $\mathcal{L}_{\text{gIoU}}(\hat{b}_i, b_j)$ proposed by Rezatofighi et al. (2019), taking into account overlap of boxes.
- An L1-loss $\mathcal{L}_{\text{prop}}$ taking into account the proportion of box width and height.
- A loss \mathcal{L}_{rel} equal to the difference between predicted and ground-truth predictions of relative distances between objects.

The following losses are not applied between matched predicted and ground-truth object pairs, but to the entire sequence of output objects at once.

A Cross-entropy Loss \mathcal{L}_{len} to the predicted number of object queries (PAR only).

A Contrastive Structural Loss to enforce in a novel way the grammatical structure found in the parse trees on the output, in our case the visual object embeddings v_k that are computed by the layout predictor and that are used to predict the object boxes and their labels.

To calculate the loss, all nodes in the parse tree, that is, leaf nodes corresponding to word tokens, and parent and root nodes corresponding to spans of word tokens, are represented separately by a positional embedding $e_j^{\text{pos}} (0 \leq j < 2N_c - 1)$ following Shiv and Quirk (2019). The positional embeddings are learned, they are agnostic of the content of the word or word spans they correspond to, and they encode the path through the tree, starting from the root, to the given node.

In a contrastive manner the loss forces the visual object representations v_k to be close to the positional embeddings e_j^{pos} , but far from those e_j^{pos} of other sentences in the minibatch. It maximizes the posterior probability that the set of visual object embeddings $V_m = \{v_k\}_k$ for sample m are matched with the set of tree positional embeddings $E_m = \{e_j^{\text{pos}}\}_j$ for the same sample m , and vice versa: $\mathcal{L}_{\text{struct}}(m) = -\log P(E_m|V_m) - \log P(V_m|E_m)$. These probabilities are computed as a softmax over similarity scores $S(m, n)$ between samples in the batch, where the denominator of the softmax sums over tree positions or objects, resp.

The similarity score for 2 samples m, n is computed as a log-sum-exp function of the cosine

similarities between the i 'th visual object v_i^m of sample m and a visually informed syntax tree context vector c_i^n representing all tree positions $\{e_j^n\}_j$ of sample n . The context vector c_i^n is computed with the attention mechanism of Bahdanau et al. (2015), with tree positional embeddings $\{e_j^n\}_j$ as keys and values, and visual embedding v_i^m as query. The dot product between query v_i^m and keys $\{e_j^n\}_j$ is first additionally normalized over the visual objects corresponding to a tree position, before the regular normalization over tree positions. These normalized dot products between keys and queries constitute a soft matching between visual objects and constituency tree node positions (note: only the positions, representing syntax and not semantics of the words, are represented by the tree positional embeddings). Since the model learns this mapping from the training signal provided by this loss, it is not necessary to manually specify which text spans are to be matched to which visual objects.

The loss has resemblance to the loss used by Xu et al. (2018), replacing their text embeddings by our visual object embeddings, and their visual embeddings by our syntax tree embeddings. Note that only the constituency parse of the input text and the output embeddings are needed. In this case, the output embeddings represent visual objects, but they are in general not confined to only represent visual objects, they could technically represent anything. Hence, the loss is not tied to layout generation in specific, and could be applied to any generation task conditioned on (grammatically) structured text, as tree positions are matched to output embeddings. This novel loss is completely independent of the text encoder and can be applied to a text encoder with explicit syntax input, or to a text encoder with implicit syntax (if a constituency parse of the input is available).⁴

The Final Loss for one training sample is the sum of the above losses \mathcal{L} , with (\hat{b}_i, b_j) matched pairs of predicted and GT objects, and each loss weighted by a different weight λ .

⁴ The loss uses explicit syntax in the form of a constituency parse, so when used to train a model with implicit syntax as input (like GPT-2_{Blip}, which does not use linearized parse trees C^{lin} as input), it adds explicit syntax information to the training signal. Nevertheless, in this study, we call such a model an "implicit syntax model with structural loss".

t_ϕ	Train set	Regime
PLM	BLLIP sents, trees	NTP
PLM _{mask}	BLLIP sents, trees	NTP
TG	BLLIP sents, trees	NTP
GPT-2 _{Blip}	BLLIP sents	NTP
GPT-2	≈8B text tokens	NTP
LLaMA	1.4T tokens	NTP

Table 1: Overview of the text encoders, their training data, and training regimes. NTP stands for next-token prediction.

$$\begin{aligned}
\mathcal{L}_{\text{total}} = & \lambda_1 \mathcal{L}_{\text{struct}} + \lambda_2 \mathcal{L}_{\text{len}} \\
& + \sum_{(i,j)} \left[\lambda_3 \mathcal{L}_{\text{label}}(\hat{b}_i, b_j) + \lambda_4 \mathcal{L}_{\text{L1}}(\hat{b}_i, b_j) \right. \\
& + \lambda_5 \mathcal{L}_{\text{IoU}}(\hat{b}_i, b_j) + \lambda_6 \mathcal{L}_{\text{prop}}(\hat{b}_i, b_j) \\
& \left. + \lambda_7 \mathcal{L}_{\text{rel}}(\hat{b}_i, b_j) \right] \tag{1}
\end{aligned}$$

3.4 Datasets

The text encoders are pretrained on datasets summarized in Table 1. We use COCO captions and instances (bounding boxes and labels; Lin et al., 2014) for training and testing the layout decoder. We use the 2017 COCO split with 118K training images and 5K validation images (both with 5 captions per image). The testing images are not usable as they have no captions and bounding box annotations. We randomly pick 5K images from the training data for validation and use the remaining 113K as training set $\mathcal{D}_{\text{train}}$. We use the 2017 COCO validation set as in-domain test set $\mathcal{D}_{\text{indom}}$. $\mathcal{D}_{\text{USCOCO}}$ is our test set of unexpected situations with 2.4K layouts and 1 caption per layout.

Collection of USCOCO We used Amazon Mechanical Turk (AMT) to collect ground-truth (caption, layout)-pairs denoting situations that are unlikely to occur in the training data. We obtained this test set in three steps. In the **first** step, we asked annotators to link sentence parts of captions in $\mathcal{D}_{\text{indom}}$ to bounding boxes.

Second, we used a script to replace linked sentence parts in the captions with a random COCO category name (o_{new} , with a different COCO supercategory than the bounding box had that the sentence part was linked to). The script also replaces the bounding box that the annotators linked to the replaced sentence part in the first step, with

a bounding box for an object of the sampled category o_{new} . We use 4 replacement strategies: the first keeps the original box merely replacing its label. The next 3 strategies also adjust the size of the box based on average dimensions of boxes with category o_{new} in $\mathcal{D}_{\text{train}}$, relative to the size of the nearest other box in the layout. The 2nd places the middle point of the new box on the middle point of the replaced box, the 3rd at an equal x -distance to the nearest object box and the 4th at an equal y -distance to the nearest object.

In the **third** step, annotators were shown the caption with the automatically replaced sentence part and the 4 corresponding automatically generated layouts. They were asked to evaluate whether the new caption is grammatically correct, and which of the 4 layouts fits the caption best (or none). Each sample of step 2 was verified by 3 different annotators. Samples where at least 2 annotators agreed on the same layout and none of the 3 annotators considered the sentence as grammatically incorrect, were added to the final USCOCO dataset.

The USCOCO test set follows a very different distribution of object categories than $\mathcal{D}_{\text{train}}$. To show this we calculate co-occurrences of object categories in all images (weighted so that every image has an equal impact) of $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{USCOCO}}$, and $\mathcal{D}_{\text{indom}}$. The co-occurrence vectors of $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{USCOCO}}$ have a cosine similarity of 47%, versus 99% for $\mathcal{D}_{\text{train}}$ and the in-domain test set $\mathcal{D}_{\text{indom}}$.

3.5 Preprocessing of the Images

Spurious Bounding Boxes (SP) Because objects annotated with bounding boxes in the COCO images are not always mentioned in the corresponding captions, we implement a filter for bounding boxes and apply it on all train and test data. The filter computes for each object class O of COCO the average diagonal length \bar{d}_O of its bounding box, over the training set, and the normalized average diagonal length \bar{d}_O^{norm} (scaled by the size of the biggest object of each image). Only the largest object of a class per image is included in these averages to limit the influence of background objects. Then, all the objects with size smaller than $0.5 \cdot \bar{d}_O$ and normalized size smaller than $0.5 \cdot \bar{d}_O^{\text{norm}}$ are discarded. The normalized threshold allows the filters to be scale invariant, while the non-normalized threshold removes fil-

tering mistakes when there is a big unimportant, unmentioned object in the image.

Crop-Pad-Normalize (CPN) To center and scale bounding boxes, we follow Collell et al. (2021). We first crop the tightest enclosing box that contains all object bounding boxes. Then, we pad symmetrically the smallest side to get a square box of height and width P . This preserves the aspect ratio when normalizing. Finally, we normalize coordinates by P , resulting in coordinates in $[0, 1]$.

3.6 Evaluation Metrics

Pr, Re, F1 precision, recall, and F1 score of predicted object labels, without taking their predicted bounding boxes into account.

Pr_{0.5}, Re_{0.5}, F1_{0.5} precision, recall, and F1 score of predicted object labels, with an Intersection over Union (IoU) threshold of 0.5 considering the areas of the predicted and ground-truth bounding boxes (Ren et al., 2017). The matching set M_{IoU} between ground-truth (GT) and predicted objects is computed in a greedy fashion based on box overlap in terms of pixels.

Re^{repl} The recall (without positions) on only the set of GT objects that have been replaced in the test set of unexpected situations $\mathcal{D}_{\text{USCOCO}}$.

Pr_{Dpw}, Re_{Dpw}, F1_{Dpw} The F1_{0.5} score penalizes an incorrect/missing label as much as it penalizes an incorrect position, while we consider an incorrect/missing label to be a worse error. Additionally, there are many plausible spatial arrangements for one caption (as explained in section 3.5 image preprocessing tries to reduce its impact). For this reason we introduce an F1 score based on the precision and recall of object pairs, penalized by the difference of the distance between the two boxes in the GT and the two boxes in the predictions. This metric penalizes incorrect positions, since a pair’s precision or recall gets downweighted when its distance is different from its distance in the GT, but it penalizes incorrect labels more, since pairs with incorrect labels have precision/recall equal to 0. Moreover, it evaluates positions of boxes relative to each other, instead of to one absolute GT layout.

First, a greedy matching set M_D between GT and predicted objects is computed based on labels and middle-point distance. Boxes \hat{b} are part of

the predicted layout L_p (with N_p object pairs), and boxes b are part of the GT layout L_{GT} (with N_{GT} object pairs). The matching function $\pi(\hat{b}_i, \hat{b}_k)$ equals 1 if predicted boxes \hat{b}_i and \hat{b}_k both have a matching box b_j and b_m in the GT (so $(i, j) \in M_D$ and $(k, m) \in M_D$), and equals 0 otherwise. D denotes Euclidean distance between box middle points, and $S_{jm,ik} \in [0, 1]$ is a normalized similarity metric based on this distance. The penalized precision Pr_{Dpw} and recall Re_{Dpw} are computed as follows:

$$S_{jm,ik} = 1 - \frac{1}{\sqrt{2}} |D(b_j, b_m) - D(\hat{b}_i, \hat{b}_k)|$$

$$\text{Pr}_{Dpw} = \frac{1}{N_p} \sum_{i \in L_p} \sum_{k \in L_p \setminus \{i\}} S_{jm,ik} \pi(\hat{b}_i, \hat{b}_k) \quad (2)$$

$$\text{Re}_{Dpw} = \frac{1}{N_{GT}} \sum_{i \in L_p} \sum_{k \in L_p \setminus \{i\}} S_{jm,ik} \pi(\hat{b}_i, \hat{b}_k) \quad (3)$$

where $(i, j) \in M_D$ and $(k, m) \in M_D$.

F1_{Dpw} is finally computed as the standard F1 of Pr_{Dpw} and Re_{Dpw} . If a sample has less than 2 boxes in the GT or predictions, respectively Re_{Dpw} or Pr_{Dpw} is undefined for that sample.⁵

4 Experiments

4.1 Experimental Set-up

All runs were repeated three times and the averages and standard deviations are reported. We used a learning rate of 10^{-4} with Adam (Kingma and Ba 2015), a batch size of 128 (64 for runs using $\mathcal{L}_{\text{struct}}$), random horizontal flips on the bounding boxes as data augmentation, and early stopping. All text encoders were frozen. Layout predictors use a hidden dimension of 256^6 and a FFN dimension of 1024, with 4 encoder layers and 6 decoder layers, and have 10M parameters. The loss weights (eq. 1) were chosen experimentally and set to $\lambda_1 \in \{0.25, 0.5, 1.0\}$, $\lambda_2 = 0.1$, $\lambda_3 = 0.5$, $\lambda_4 = 5$, $\lambda_5 = 2$, $\lambda_6 = 0.5$, $\lambda_7 = 0.5$. We took most of the other PAR hyperparameters from Carion et al. (2020).

We run all text encoders with the smallest GPT-2 architecture (125M params), for which

⁵There are fewer than 2 boxes for 0% of samples in $\mathcal{D}_{\text{USCOCO}}$ and 19% in $\mathcal{D}_{\text{indom}}$. Re and $\text{Re}_{0.5}$ are defined for samples with only 1 box, and samples with 0 boxes almost do not occur.

⁶We use the same dimension regardless of the text encoder to allow for a fair comparison. Increasing the dimension did not improve results.

we reuse checkpoints shared by Qian et al. (2021) for PLM, PLM_{mask} and $\text{GPT-2}_{\text{Bllip}}$. We also run $\text{GPT-2-1g}_{\text{Bllip}}$, GPT-2-1g and TG-1g with the larger GPT-2 architecture (755M params). GPT-2 and GPT-2-1g runs use checkpoints from HuggingFace (Wolf et al., 2020), and LLaMA runs use checkpoints shared by Meta. We train $\text{GPT-2-1g}_{\text{Bllip}}$ ourselves, using the code of Qian et al. (2021).

Models were trained on one 16GB Tesla P100 or 32GB Tesla V100 GPU (except the LLaMA-33B runs which were trained on a 80GB A100).

Training TG We train TG and TG-1g like PLM and baseline $\text{GPT-2}_{\text{Bllip}}$ following Qian et al. (2021), with a learning rate 10^{-5} , the AdamW optimizer, a batch size of 5, and trained until convergence on the development set of the BLLIP_{LG} dataset split (Charniak et al., 2000). We implement TG with the recursive masking procedure of Sartran et al. (2022), but without the relative positional encodings, since these do not contribute much to the performance, and because GPT-2 uses absolute position embeddings.

5 Results and Discussion

5.1 Layout Prediction

5.1.1 Preprocessing of Images

We ran a comparison of preprocessing for the PLM and $\text{GPT-2}_{\text{Bllip}}$ text encoders (both using PAR). All conclusions were identical.

Using SP gives small but significant improvements in $\text{F1}_{0.5}$ and F1 on both test sets, and larger improvements when also normalizing bounding boxes with CPN. Using CPN increases the position sensitive $\text{F1}_{0.5}$ metric drastically on both test sets, even more so when also using SP. In a human evaluation with AMT, annotators chose the best layout given a COCO caption from $\mathcal{D}_{\text{indom}}$. A total of 500 captions with 2 corresponding layouts (one from $\mathcal{D}_{\text{indom}} + \text{CPN} + \text{SP}$ and one from $\mathcal{D}_{\text{indom}} + \text{CPN}$) were evaluated by 3 annotators, who preferred SP in 37% of cases, as opposed to 18.6% where they preferred not using SP (44.4% of the time they were indifferent). These results suggest that the preprocessing techniques improve the alignment of COCO bounding boxes with their captions, and that the best alignment is achieved when using both.

	$F1_{0.5} \uparrow$	$\mathcal{D}_{\text{indom}}$ $F1 \uparrow$	$F1_{Dpw} \uparrow$	$F1_{0.5} \uparrow$	$\mathcal{D}_{\text{USCOCO}}$ $F1 \uparrow$	$F1_{Dpw} \uparrow$
ObjLSTM*	.185 \pm .021	.676 \pm .006	.356 \pm .019	.099 \pm .013	.524 \pm .009	.16 \pm .019
ObjLSTM _{lrg} + GPT-2 _{Bllip}	.104 \pm .003	.542 \pm .01	.238 \pm .013	.074 \pm .003	.404 \pm .014	.078 \pm .007
ObjLSTM _{lrg} + GPT-2 _{Bllip}	.167 \pm .005	.65 \pm .006	.345 \pm .01	.1 \pm .003	.524 \pm .016	.174 \pm .019
SEQ + GPT-2 _{Bllip}	.271 \pm .004	.597 \pm .01	.304 \pm .011	.167 \pm .001	.485 \pm .006	.149 \pm .007
PAR + GPT-2 _{Bllip}	.296 \pm .004	.67 \pm .014	.375 \pm .018	.18 \pm .001	.576 \pm .026	.229 \pm .036
SEQ + TG	.28 \pm .002	.638 \pm .006	.344 \pm .011	.177 \pm .002	.541 \pm .002	.203 \pm .004
PAR + TG	.306 \pm .008	.69 \pm .002	.398 \pm .008	.185 \pm .004	.6 \pm .004	.255 \pm .005

Table 2: PAR, SEQ, and ObjLSTM (baseline) layout predictor results on USCOCO and $\mathcal{D}_{\text{indom}}$, incl. $F1$, $F1_{0.5}$ and $F1_{Dpw}$ showing that the PAR decoding model performs best. All entries use the GPT-2_{Bllip} or TG text encoder (without structural loss), except for ObjLSTM* which uses a multimodal text encoder trained on images and text (Li et al., 2019c; Xu et al., 2018). ObjLSTM_{lrg} is scaled up to same number of parameters as SEQ and PAR, and uses a layout predictor with a transformer encoder before the LSTM decoder, like SEQ and PAR. Best results of models using implicit syntax (upper rows) and those with explicit syntax (lower rows, with TG) are marked in bold.

5.1.2 Layout Prediction Models

Table 2 compares our new PAR and SEQ layout predictors with the ObjLSTM baseline. All models use either the GPT-2_{Bllip} or TG text encoder (based on the small GPT-2 architecture), except for ObjLSTM* which uses a multimodal text encoder following Li et al. (2019c) and Xu et al. (2018). The \mathcal{L}_{rel} and $\mathcal{L}_{\text{prop}}$ losses are used for the SEQ and PAR runs (in Table 2 and subsequent tables). These losses give minor consistent improvements in $F1_{0.5}$, while keeping $F1$ more or less constant.

Both SEQ / PAR + GPT-2_{Bllip} models outperform all ObjLSTM baselines by significant margin on the position sensitive $F1_{0.5}$ metric on both test sets (even though ObjLSTM* uses a text encoder that has been pretrained on multimodal data). PAR + GPT-2_{Bllip} obtains better $F1_{Dpw}$ and position insensitive $F1$ scores than baselines on the unexpected test set, and similar $F1_{Dpw}$ and $F1$ on $\mathcal{D}_{\text{indom}}$. SEQ + GPT-2_{Bllip} lags a bit behind on the last 2 metrics.

PAR obtains a significantly better precision than SEQ, both with and without object positions (Pr and $Pr_{0.5}$), on both test sets, both with GPT-2_{Bllip} and TG, resulting in greater $F1$ scores. This could be attributed to the fact that the n^{th} prediction with SEQ is conditioned only on the text and $n - 1$ preceding objects, while with PAR, all predictions are conditioned on the text and on all other objects. The fact that for generating language, autoregressive models like SEQ are su-

perior to non-autoregressive models like PAR, but vice versa for generating a set of visual objects, may be due to the inherent sequential character of language, as opposed to the set of visual objects in a layout, which does not follow a natural sequential order. When generating a set of objects in parallel, the transformer’s self-attention can model all pairwise relationships between objects before assigning any positions or labels. In contrast, when modeling a sequence of objects autoregressively, the model is forced to decide on the first object’s label and position without being able to take into account the rest of the generated objects, and it cannot change those decisions later on.

Since the PAR model scores higher and is more efficient (it decodes all \hat{b}_i in one forward pass, compared to one forward pass per \hat{b}_i for SEQ), we use PAR in subsequent experiments.

5.2 Improved Generalization to USCOCO Data with Syntax

5.2.1 Explicitly Modeling Syntax

Table 3 shows layout prediction $F1_{0.5}$, $F1$, and $F1_{Dpw}$ on the USCOCO test set of PAR with implicitly structured GPT-2_{Bllip}, GPT-2, and LLaMA-7B text encoders (upper half) vs. with explicitly structured PLM, PLM_{mask} and TG text encoders (bottom half), with (rows with + $\mathcal{L}_{\text{struct}}$) and without ($\lambda_1 = 0$) structural loss.

Without structural loss, all smaller 125M models achieve very similar $F1_{0.5}$ scores compared

		Size	$F1_{Dpw} \uparrow$	\mathcal{D}_{USCOCO} $F1_{0.5} \uparrow$	$F1 \uparrow$
GPT-2		125M	.207 \pm .019	.179 \pm .008	.566 \pm .02
GPT-2	+ \mathcal{L}_{struct}		.187 \pm .006	.184 \pm .007	.555 \pm .011
GPT-2 _{Bllip}			.229 \pm .036	.18 \pm .001	.576 \pm .026
GPT-2 _{Bllip}	+ \mathcal{L}_{struct}		.233 \pm .014	.192 \pm .003	.574 \pm .014
GPT-2-1g		755M	.283 \pm .047	.188 \pm .005	.61 \pm .03
GPT-2-1g	+ \mathcal{L}_{struct}		.292 \pm .025	.205 \pm .009	.628 \pm .016
GPT-2-1g _{Bllip}			.233 \pm .027	.183 \pm .002	.586 \pm .019
GPT-2-1g _{Bllip}	+ \mathcal{L}_{struct}		.234 \pm .019	.196 \pm .005	.579 \pm .006
LLaMA-7B		7B	.231 \pm .014	.179 \pm .007	.583 \pm .011
LLaMA-7B	+ \mathcal{L}_{struct}		.26 \pm .026	.192 \pm .01	.602 \pm .02
PLM		125M	.226 \pm .006	.18 \pm .002	.579 \pm .002
PLM	+ \mathcal{L}_{struct}		.282 \pm .048	.192 \pm .002	.61 \pm .033
PLM _{mask}			.234 \pm .012	.176 \pm .005	.588 \pm .01
PLM _{mask}	+ \mathcal{L}_{struct}		.28 \pm .039	.191 \pm .007	.612 \pm .024
TG			.255 \pm .005	.185 \pm .004	.6 \pm .004
TG	+ \mathcal{L}_{struct}		.318 \pm .026	.192 \pm .008	.641 \pm .018
TG-1g		755M	.283 \pm .017	.183 \pm .008	.621 \pm .014
TG-1g	+ \mathcal{L}_{struct}		.327 \pm .018	.195 \pm .006	.645 \pm .01

Table 3: Text encoders with implicit (above double line) and explicit (below double line) syntax, and structural loss results on USCOCO: F1, precision and recall, with and without IoU threshold and pairwise distance weighted. All entries use the PAR layout predictor. Results of the best (in terms of $F1_{Dpw}$) λ_1 for each model type are shown.

to the baseline GPT-2_{Bllip}, and only TG is able to slightly improve the F1 and $F1_{Dpw}$ scores. We assume that models with explicit syntax, i.e., that integrate syntax in the input sentence, do not learn to fully utilize the compositionality of the syntax with current learning objectives.

We observe a noticeable increase over all metrics by using GPT-2-1g compared to GPT-2 which is to be expected, while GPT-2-1g_{Bllip} and GPT-2_{Bllip} perform equally where we assume that the training with a relatively small dataset does not fully exploit the capabilities of a larger model. TG-1g obtains similar scores as TG with a small increase for the $F1_{Dpw}$ and performs on par with GPT-2-1g while trained on only a fraction of the data.

Notable is that the very large LLaMA-7B model performs on par with GPT-2_{Bllip} and GPT-2-1g_{Bllip}. A possible explanation could be the quite drastic downscaling of the 4096-dimensional features of LLaMA-7B to 256 dimensions for our layout predictor by a linear layer. Using a 4096-dimensional hidden dimension for the layout predictor did not improve results. This increased

the number of trainable parameters by two orders of magnitude, and the large resulting model possibly overfitted the COCO train set.

TG outperforms PLM and PLM_{mask} in F1 and $F1_{Dpw}$, which proves that restricting the attention masking scheme to follow a recursive pattern according to the recursion of syntax in the sentence helps generalizing to unexpected situations. This is in line with Sartran et al. (2022) who find that TG text encoders show better syntactic generalization.

5.2.2 Structural Loss

Table 3 displays in the rows with + \mathcal{L}_{struct} the impact of training with our structural loss function, with the best weight λ_1 in the total loss in eq. (1) chosen from {0.25, 0.5, 1.0}. $F1_{0.5}$ and $Pr_{0.5}$ slightly increase for all models and all λ_1 values. $Re_{0.5}$ is minimally affected, except for some explicit syntax models and LLaMA that see a slight increase.

F1, $F1_{Dpw}$ For implicit syntax models, with increasing λ_1 , Re and Re_{Dpw} decrease severely.

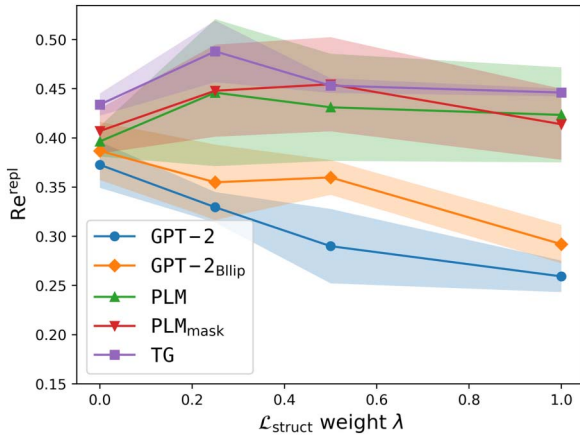


Figure 3: Recall on replaced objects Re^{repl} in USCOCO vs. structural loss $\mathcal{L}_{\text{struct}}$ weight λ_1 .

Pr and Pr_{Dpw} first increase and then decrease again, resulting in sometimes stable but eventually decreasing F1 scores. For models with explicit syntax, Re and Re_{Dpw} increase most for small loss weights ($\lambda_1 = 0.25$), while Pr and Pr_{Dpw} top at $\lambda_1 = 0.5$ or 1.0 . Together this causes an improvement in F1 but mainly a sharp rise in F1_{Dpw} . These trends are more prominent for small models, but persist for large models, incl. LLaMA. F1_{Dpw} peaks for TG/TG-1g with $0.25 \cdot \mathcal{L}_{\text{struct}}$ at $0.318/0.327$, which is a $\approx 40\%$ increase of the baselines' performance of GPT-2-Bllip/GPT-2-1g_{Bllip} at $0.229/0.233$.

The loss, enforcing explicit constituency tree structure in the output visual embeddings, trains the layout predictor to not lose the explicit structure encoded by TG, PLM and PLM_{mask}. This compositional structure causes a disentangled, recursive representation of visual scenes (Hawkings, 2021; Hauser et al., 2002), facilitating the replacement of objects with unexpected different objects for input sentences that contain unexpected combinations of objects. For models with implicit syntax, the loss tries to enforce a structure that is not explicitly available in the models' input (as opposed to for the models with explicit syntax), which may lead to a more difficult learning objective.

Re^{repl} Figure 3 shows the recall Re^{repl} on the replaced object of USCOCO (the unusual object). Re^{repl} increases for models with explicit syntax, topping at $\lambda_1 = 0.25$, while it decreases sharply for models with implicit syntax GPT-2 and GPT-2-Bllip.

5.2.3 Overview of Explicit vs. Implicit Syntax

Table 4 gathers results on both test sets.⁷ The results for all models, but most notably the implicit syntax models, drop significantly on USCOCO compared to $\mathcal{D}_{\text{indom}}$, confirming that current state-of-the-art models struggle with generating unexpected visual scenes.

$\mathcal{D}_{\text{indom}}$ Small models that explicitly model syntax obtain slightly better results than the small baseline models for all metrics. Models with implicit syntax might perform well on in-domain test data because they have memorized the common structures in training data. The large models that were pretrained on huge text datasets (GPT-2-1g, LLaMA-xB) outperform TG-1g on $\mathcal{D}_{\text{indom}}$, showing that their pretraining does help for this task, but the drop in USCOCO scores suggests that they might overfit the memorized patterns. Situations described in COCO captions are commonly found in pretraining data, so that syntax is not needed to predict their visual layouts. The unexpected USCOCO situations however require the extra compositionality offered by explicit syntax.

USCOCO We clearly see improvement in results of models that explicitly model syntax, showing the generalization capabilities needed to perform well on the unseen object combinations of USCOCO, provided it is enforced by a correct learning objective as discussed in section 5.2.2. This increase comes without a decrease in performance on the in-domain test data. This is important because it will lead to efficient models for natural language processing that can generalize to examples not seen in the training data, exploiting compositionality.

GPT-2^{shuffle}_{Bllip} Another indication that the models that implicitly model syntax do not use the structure of natural language to the same extent, but rather exploit co-occurrences in training data, is the fact that GPT-2^{shuffle}_{Bllip}, which is trained to generate layouts from sentences with shuffled words, achieves only slightly worse results than GPT-2_{Bllip}, even on the position sensitive $\text{F1}_{0.5}$ and F1_{Dpw} metrics.

⁷Although CLIP (Radford et al., 2021) has been pretrained on multimodal data, and the other text encoders were not (ignoring for a moment the ObjLSTM baseline), we tested CLIP's sentence embedding, but results were poor.

	Size	$\mathcal{D}_{\text{indom}}$			$\mathcal{D}_{\text{USCOCO}}$		
		$F1_{0.5} \uparrow$	$F1 \uparrow$	$F1_{Dpw} \uparrow$	$F1_{0.5} \uparrow$	$F1 \uparrow$	$F1_{Dpw} \uparrow$
GPT-2 ^{shuffle} _{Bllip}	125M	.286 ± .006	.656 ± .005	.349 ± .012	.166 ± .002	.566 ± .003	.213 ± .004
GPT-2	125M	.294 ± .004	.66 ± .01	.353 ± .019	.179 ± .008	.566 ± .02	.207 ± .019
GPT-2 _{Bllip}		.296 ± .004	.67 ± .014	.375 ± .018	.18 ± .001	.576 ± .026	.229 ± .036
GPT-2-1g	755M	.308 ± .001	.702 ± .007	.414 ± .013	.188 ± .005	.61 ± .03	.283 ± .047
GPT-2-1g _{Bllip}		.298 ± .004	.676 ± .01	.38 ± .013	.183 ± .002	.586 ± .019	.233 ± .027
LLaMA-7B	7B	.306 ± .001	.701 ± .003	.411 ± .008	.179 ± .007	.583 ± .011	.231 ± .014
LLaMA-33B	33B	.305 ± .005	.699 ± .003	.406 ± .002	.181 ± .006	.577 ± .008	.225 ± .011
TGRB	125M	.299 ± .005	.683 ± .01	.391 ± .015	.178 ± .004	.571 ± .011	.216 ± .016
TGRB + $\mathcal{L}_{\text{struct}}$.3 ± .005	.67 ± .01	.358 ± .012	.189 ± .007	.606 ± .017	.278 ± .02
PLM + $\mathcal{L}_{\text{struct}}$	125M	.301 ± .006	.677 ± .022	.378 ± .038	.192 ± .002	.61 ± .033	.282 ± .048
PLM _{mask} + $\mathcal{L}_{\text{struct}}$.3 ± .004	.683 ± .003	.388 ± .007	.191 ± .007	.612 ± .024	.28 ± .039
TG + $\mathcal{L}_{\text{struct}}$.305 ± .005	.685 ± .012	.379 ± .028	.192 ± .008	.641 ± .018	.318 ± .026
TG-1g + $\mathcal{L}_{\text{struct}}$	755M	.306 ± .004	.692 ± .002	.392 ± .007	.195 ± .006	.645 ± .01	.327 ± .018

Table 4: Final model results in terms of $F1$, $F1_{0.5}$, and $F1_{Dpw}$ on the in-domain (left) and USCOCO (right) test sets. All entries use the PAR layout predictor, and explicit syntax models use $\lambda_1 = 0.25$ or $\lambda_1 = 0.50$. GPT-2(-1g) and LLaMA have been pretrained on much larger text datasets.

TGRB obtains similar scores to TG on in-domain data with trivial right-branching trees. On unexpected data, performance drops, proving the importance of syntax especially for generalizing to unexpected data. The structured loss improves generalization shown by USCOCO results, but not to the same extent as for TG.⁸

5.2.4 Human Evaluation

Proper automatic evaluation of performance on the text-to-layout prediction task is hard, since potentially many spatial layouts may fit the scene described by a sentence. Our metrics compare the predictions to one single ground-truth, ignoring this fact, so we used AMT for a human evaluation of predicted layouts for 500 randomly sampled USCOCO captions. For each caption, 3 annotators chose the layout that best fit the caption from a pair of two, based on the following criteria: Whether the layout displays all objects present in the caption, whether the objects’ spatial arrangement corresponds to the caption, whether the objects have reasonable proportions and finally whether object predictions that are not explicitly

⁸It is not surprising that performance is partially retained with right-branching trees, since English has a right-branching tendency: The $F1$ overlap between the trivial constituency trees and the silver-truth trees for COCO validation captions is 0.62. Further, the constituency tags (e.g., ‘NP’, ‘PP’) are still included, and N_{fin} syntax tokens are added to the N_c caption tokens, granting the model more processing power.

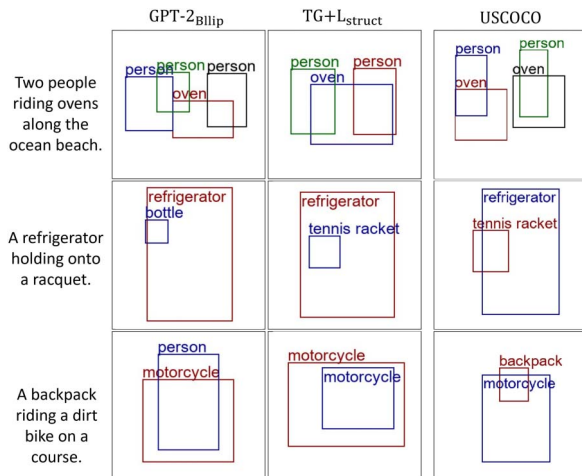


Figure 4: Examples of generated layouts where annotators chose the layout of TG + $\mathcal{L}_{\text{struct}}$ over the layout of GPT-2_{Bllip} (first 2 examples) and vice versa (last example).

mentioned in the caption do fit the rest of the scene (i.e., the layout should not contain any absurd or unexpected objects that are not explicitly mentioned in the caption). Figure 4 shows some examples of generated layouts and the annotators’ decisions.

The results in Figure 5 are in line with the quantitative results where our structural loss proved beneficial for TG. This confirms that explicit structure does not improve layout prediction of unexpected combinations by itself, but together with our structural loss it causes a significant improvement. We calculated the agreement of the

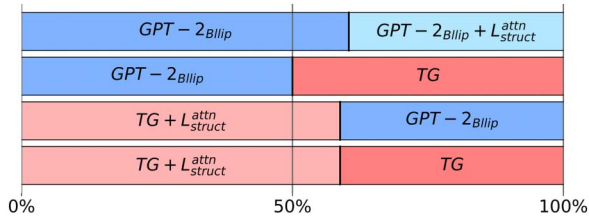


Figure 5: Human evaluation of generated layouts by GPT-2_{Blipip} (+ \mathcal{L}_{struct}) and TG (+ \mathcal{L}_{struct}) on USCOCO. Annotators choose the best layout between 2 layouts (anonymized and order-randomized).

human evaluation with our quantitative metrics, and found 15.2% for $F1_{0.5}$, 41.7% for F1 and 42.5% for $F1_{Dpw}$.⁹ This confirms the previously mentioned suspicion that $F1_{0.5}$ is far less suitable for the evaluation of layout generation than F1 and $F1_{Dpw}$.

5.2.5 Constituency Tree Probes

To test how the loss affects syntax information in the text embeddings, we run a classifier probe inspired by Tenney et al. (2019) on the text encoder output and subsequent layers of the encoder of the layout prediction model. The probe classifies random spans of tokens as being a constituent or not (but ignores their tag).

Figure 6 shows that all text encoders’ outputs (probe layer 0) get good F1 scores and hence do encode syntax. Without the proposed structural loss, probing results quickly deteriorate in subsequent layers, presumably because the encoder has too little incentive to use and retain constituency structure, because COCO training data contains only situations common in pretraining data and requires no syntactical reasoning. The figure also shows that it is easier to predict constituency structure from outputs of text encoders with explicit syntax than of those with implicit syntax, which is not surprising because of the former’s pretraining and the presence of parentheses and tags.

The structural loss helps to almost perfectly retain the constituency structure. The loss matches the output (in our case visual objects) to constituency tree positions, and as the probe shows, to do so, it propagates the constituency tree information present in the text through the model. For GPT-2_{Blipip}, except for an initial drop caused by the linear projection to a lower dimension, the

⁹The low percentages were to be expected since metrics often rank layouts equally (when both layouts obtain the same score), while annotators were not given that option.

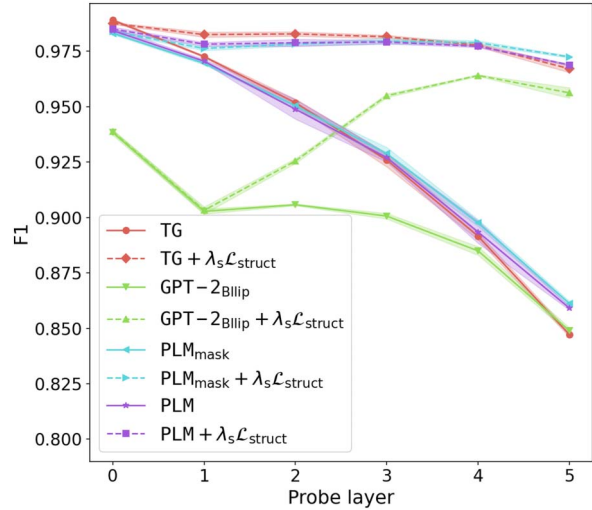


Figure 6: Constituency tree probe results for GPT-2_{Blipip}, TG, PLM, and PLM_{mask} on USCOCO. Layer 0 corresponds to text encoder output (with model-dependent dimensions), layer 1 to the 256-dimensional embedding after linear projection, layer 2 to the output of the 1st encoder layer, etc.

loss improves probing F1 in later layers, even beyond the F1 for raw text encoder output. That this increase does not lead to improved layout predictions could be explained by the relevant syntax being encoded in a different, more implicit form that is harder for downstream models to learn to use.

5.2.6 Computational Cost

The addition of structural (parenthesis and tag) tokens to explicit syntax model input causes the number of tokens $N_c + N_{lin}$ to be larger than the number of tokens N_c that implicit models use to encode the same sentence. GPT-2 needs only 11 tokens on average per sentence in the COCO validation set, versus TG that needs 38 and PLM that needs 30. This translates in a greater computational cost for the explicit syntax models.

Nevertheless, the small TG + \mathcal{L}_{struct} , pre-trained only on BLLIP_{LG}, outperforms the large GPT-2-1g that has been pre-trained on a much larger dataset. This entails multiple computational advantages: smaller memory footprint and fewer resources and less time needed for pretraining.

6 Limitations & Future Work

One limitation of layout decoding with explicitly structured language models is the reliance upon a syntax parsing model to obtain the constituency trees for input captions. While syntax parsing

models have shown very high performance (the parser of Kitaev and Klein [2018] obtains an F1 score of 95.13 on the Penn Treebank [Marcus et al., 1993], which contains longer and more syntactically complex sentences than typical COCO captions), grammatical errors in the used prompts might result in incorrect parses and hence in worse layout generations, compared to language models without explicit syntax (that do not need a parser). We leave an investigation into this phenomenon for further research. However, we do note an increased performance of layout generation even with only trivial right-branching trees over implicit syntax models (visible in Table 4), which might be an indication of robustness against grammatical errors for models that explicitly encode syntax.

Furthermore, while we show that explicitly modeling syntax improves layout prediction for absurd situations, this out-of-distribution generation task still remains difficult even for the best layout predictor models: there is a 35%–37% drop in F1 score and 17%–26% drop in $F1_{Dpw}$ on USCOCO compared to the in-domain test set. The introduction of USCOCO allows further research to evaluate new layout generation models on their out-of-distribution and absurd generation capabilities.

Very recent work has prompted the GPT-4 API to generate SVG or TikZ code that can be rendered into schematic images, which can then be used to guide the generation of more detailed images (Bubeck et al., 2023; Zhang et al., 2023). The layout prediction models discussed in our paper generate bounding box coordinates and class labels, which are hard to directly compare to code or rendered images. Moreover, we studied the role that explicit grammar can play for robustness with respect to absurd inputs, which would not have been possible with the GPT-4 API. However, using LLMs for layout prediction can be a promising direction for future work.

7 Conclusion

We evaluated models that implicitly and explicitly capture the syntax of a sentence and assessed how well they retain this syntax in the representations when performing the downstream task of layout prediction of objects on a 2D canvas. To test compositional understanding, we collected a test set of grammatically correct sentences and layouts

describing compositions of entities and relations that unlikely have been seen during training. We introduced a novel parallel decoder for layout prediction based on a transformer architecture, but most importantly we proposed a novel contrastive structural loss that enforces the encoding of syntax structure in the representation of a visual scene and show that it increases generalization to unexpected compositions resulting in large performance gains in the task of 2D spatial layout prediction conditioned on text. The loss has the potential to be used in other generation tasks that condition on structured input, which could be investigated in future work. Our research is a step forward in retaining structured knowledge in neural models.

Acknowledgments

This work is part of the CALCULUS project, which is funded by the ERC advanced grant H2020-ERC-2017 ADG 788506.¹⁰ It also received funding from the Research Foundation – Flanders (FWO) under grant agreement no. G078618N. The computational resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by FWO and the Flemish Government. We thank the reviewers and action editors of the TACL journal for their insightful feedback and comments.

References

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR*. <https://doi.org/10.48550/arXiv.1409.0473>
- Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott M. Lundberg, Harsha Nori, Hamid Palangi, Marco Túlio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with GPT-4. *CoRR*, abs/2303.12712. <https://doi.org/10.48550/arXiv.2303.12712>
- Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end

¹⁰<https://calculus-project.eu/>.

- object detection with transformers. In *Computer Vision - ECCV 2020 - 16th European Conference*, volume 12346 of *Lecture Notes in Computer Science*, pages 213–229. Springer. https://doi.org/10.1007/978-3-030-58452-8_13
- Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson. 2000. Bllip 1987-89 wsj corpus release 1 ldc2000t43. <https://doi.org/10.35111/fwew-da58>
- Minghao Chen, Iro Laina, and Andrea Vedaldi. 2023. Training-free layout control with cross-attention guidance. *CoRR*, abs/2304.03373. <https://doi.org/10.48550/arXiv.2304.03373>
- Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 2331–2336. The Association for Computational Linguistics. <https://doi.org/10.18653/v1/d16-1257>
- Noam Chomsky. 1965. *Aspects of the Theory of Syntax*. MIT Press. <https://doi.org/10.21236/AD0616323>
- Guillem Collell, Thierry Deruyttere, and Marie-Francine Moens. 2021. Probing spatial clues: Canonical spatial templates for object relationship understanding. *IEEE Access*, 9:134298–134318. <https://doi.org/10.1109/ACCESS.2021.3113781>
- Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. 2022. Diffedit: Diffusion-based semantic image editing with mask guidance. *CoRR*, abs/2210.11427. <https://doi.org/10.48550/arXiv.2210.11427>
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*, pages 4171–4186. Association for Computational Linguistics. <https://doi.org/10.18653/v1/n19-1423>
- Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 199–209. The Association for Computational Linguistics. <https://doi.org/10.18653/v1/n16-1024>, PubMed: 26993434
- Steven M. Frankland and Joshua D. Greene. 2020. Concepts and compositionality: In search of the brain’s language of thought. *Annual Review of Psychology*, 71(1):273–303. <https://doi.org/10.1146/annurev-psych-122216-011829>, PubMed: 31550985
- Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. 2020. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673. <https://doi.org/10.1038/s42256-020-00257-z>
- Tejas Gokhale, Hamid Palangi, Besmira Nushi, Vibhav Vineet, Eric Horvitz, Ece Kamar, Chitta Baral, and Yezhou Yang. 2022. Benchmarking spatial relationships in text-to-image generation. *CoRR*, abs/2212.10015. <https://doi.org/10.48550/arXiv.2212.10015>
- Marc D. Hauser, Noam Chomsky, and W. Tecumseh Fitch. 2002. The faculty of language: What is it, who has it, and how did it evolve? *Science*, 298(5598):1569–1579. <https://doi.org/10.1126/science.298.5598.1569>, PubMed: 12446899
- Jeff Hawkins. 2021. *A Thousand Brains: A New Theory of Intelligence*. Basic Books.
- John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 4129–4138. Association for Computational Linguistics. <https://doi.org/10.18653/v1/n19-1419>
- Seunghoon Hong, Dingdong Yang, Jongwook Choi, and Honglak Lee. 2018. Inferring semantic layout for hierarchical text-to-image

- synthesis. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, pages 7986–7994. IEEE Computer Society. <https://doi.org/10.1109/CVPR.2018.00833>
- Wilhelm Humboldt. 1999. *On Language: On the Diversity of Human Language Construction and its Influence on the Mental Development of the Human Species*. Cambridge University Press.
- Justin Johnson, Agrim Gupta, and Li Fei-Fei. 2018. Image generation from scene graphs. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 1219–1228. Computer Vision Foundation / IEEE Computer Society. <https://doi.org/10.1109/CVPR.2018.00133>
- Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR*. <https://doi.org/10.48550/arXiv.1412.6980>
- Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL*, pages 3499–3505. Association for Computational Linguistics. <https://doi.org/10.18653/v1/p19-1340>
- Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 2676–2686. Association for Computational Linguistics. <https://doi.org/10.18653/v1/P18-1249>
- Harold W. Kuhn. 2010. The Hungarian method for the assignment problem. In Michael Jünger, Thomas M. Lieblich, Denis Naddef, George L. Nemhauser, William R. Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A. Wolsey, editors, *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*, Springer, pages 29–47. https://doi.org/10.1007/978-3-540-68279-0_2
- Artur Kulmizev and Joakim Nivre. 2021. Schrödinger’s tree - on syntax and neural language models. *CoRR*, abs/2110.08887. <https://doi.org/10.3389/frai.2022.796788>, PubMed: 36325030
- Hsin-Ying Lee, Lu Jiang, Irfan Essa, Phuong B. Le, Haifeng Gong, Ming-Hsuan Yang, and Weilong Yang. 2020. Neural design network: Graphic layout generation with constraints. In *Computer Vision - ECCV*, volume 12348 of *Lecture Notes in Computer Science*, pages 491–506. Springer. https://doi.org/10.1007/978-3-030-58580-8_29
- Boren Li, Boyu Zhuang, Mingyang Li, and Jian Gu. 2019a. Seq-sg2sl: Inferring semantic layout from scene graph through sequence to sequence learning. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV*, pages 7434–7442. IEEE. <https://doi.org/10.1109/ICCV.2019.00753>
- Jianan Li, Jimei Yang, Aaron Hertzmann, Jianming Zhang, and Tingfa Xu. 2019b. Layoutgan: Generating graphic layouts with wireframe discriminators. In *7th International Conference on Learning Representations, ICLR*. OpenReview.net.
- Wenbo Li, Pengchuan Zhang, Lei, Qiuyuan Huang, Xiaodong He, Siwei Lyu, and Jianfeng Gao. 2019c. Object-driven text-to-image synthesis via adversarial training. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019*, pages 12174–12182. Computer Vision Foundation / IEEE.
- 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 *Computer Vision - ECCV 2014 - 13th European Conference*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer. https://doi.org/10.1007/978-3-319-10602-1_48
- Tal Linzen and Marco Baroni. 2021. Syntactic structure from deep learning. *Annual Review of Linguistics*, 7(1):195–212. <https://doi.org/10.1146/annurev-linguistics-032020-051035>
- Christopher D. Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. 2020. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Science*,

- USA, 117(48):30046–30054. <https://doi.org/10.1073/pnas.1907367117>, PubMed: 32493748
- 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. <https://doi.org/10.21236/ADA273556>
- Letitia Parcalabescu, Michele Cafagna, Lilitta Muradjan, Anette Frank, Iacer Calixto, and Albert Gatt. 2022. VALSE: A task-independent benchmark for vision and language models centered on linguistic phenomena. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL, pages 8253–8280. Association for Computational Linguistics.
- Laura Pérez-Mayos, Miguel Ballesteros, and Leo Wanner. 2021. How much pretraining data do language models need to learn syntax? In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 1571–1582. Association for Computational Linguistics. <https://doi.org/10.18653/v1/2021.emnlp-main.118>
- Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 2227–2237. Association for Computational Linguistics. <https://doi.org/10.18653/v1/n18-1202>
- Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. 2022. Dreamfusion: Text-to-3d using 2d diffusion. *CoRR*, abs/2209.14988. <https://doi.org/10.48550/arXiv.2209.14988>
- Diana Nicoleta Popa, Julien Perez, James Henderson, and Eric Gaussier. 2021. Towards syntax-aware token embeddings. *Natural Language Engineering*, 27(6):691–720. <https://doi.org/10.1017/S1351324920000297>
- Peng Qian, Tahira Naseem, Roger Levy, and Ramón Fernández Astudillo. 2021. Structural guidance for transformer language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP*, pages 3735–3745. Association for Computational Linguistics. <https://doi.org/10.18653/v1/2021.acl-long.289>
- Leigang Qu, Shengqiong Wu, Hao Fei, Liqiang Nie, and Tat-Seng Chua. 2023. Layoutllm-t2i: Eliciting layout guidance from llm for text-to-image generation. <https://doi.org/10.48550/arXiv.2308.05095>
- Gorjan Radevski, Guillem Collell, Marie-Francine Moens, and Tinne Tuytelaars. 2020. Decoding language spatial relations to 2d spatial arrangements. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16–20 November 2020*, pages 4549–4560. Association for Computational Linguistics. <https://doi.org/10.18653/v1/2020.findings-emnlp.408>
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.
- Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with CLIP latents. *CoRR*, abs/2204.06125. <https://doi.org/10.48550/arXiv.2204.06125>
- Royi Rassin, Shauli Ravfogel, and Yoav Goldberg. 2022. DALLE-2 is seeing double: Flaws in word-to-concept mapping in text2image models. In *Proceedings of the Fifth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2022*, pages 335–345.

- Association for Computational Linguistics. <https://doi.org/10.18653/v1/2022.blackboxnlp-1.28>
- Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. 2017. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149. <https://doi.org/10.1109/TPAMI.2016.2577031>, PubMed: 27295650
- Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian D. Reid, and Silvio Savarese. 2019. Generalized intersection over union: A metric and a loss for bounding box regression. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 658–666. Computer Vision Foundation / IEEE. <https://doi.org/10.1109/CVPR.2019.00075>
- David Samuel, Andrey Kutuzov, Lilja Øvrelid, and Erik Velldal. 2023. Trained on 100 million words and still in shape: BERT meets British National Corpus. In *Findings of the Association for Computational Linguistics: EAACL*, pages 1909–1929. Association for Computational Linguistics. <https://doi.org/10.18653/v1/2023.findings-eacl.146>
- Laurent Sartran, Samuel Barrett, Adhiguna Kuncoro, Milos Stanojevic, Phil Blunsom, and Chris Dyer. 2022. Transformer grammars: Augmenting transformer language models with syntactic inductive biases at scale. *Transactions of the Association for Computational Linguistics*, 10:1423–1439. https://doi.org/10.1162/tacl_a_00526
- Vighnesh Leonardo Shiv and Chris Quirk. 2019. Novel positional encodings to enable tree-based transformers. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS*, pages 12058–12068.
- Russell Stewart, Mykhaylo Andriluka, and Andrew Y. Ng. 2016. End-to-end people detection in crowded scenes. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 2325–2333. IEEE Computer Society. <https://doi.org/10.1109/CVPR.2016.255>
- Fuwen Tan, Song Feng, and Vicente Ordonez. 2018. Text2scene: Generating abstract scenes from textual descriptions. *CoRR*, abs/1809.01110. <https://doi.org/10.48550/arXiv.1809.01110>
- Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? Probing for sentence structure in contextualized word representations. In *7th International Conference on Learning Representations, ICLR*. OpenReview.net.
- Tristan Thrush, Ryan Jiang, Max Bartolo, Amanpreet Singh, Adina Williams, Douwe Kiela, and Candace Ross. 2022. Winoground: Probing vision and language models for visiolinguistic compositionality. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 5228–5238. IEEE. <https://doi.org/10.1109/CVPR52688.2022.00517>
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971. <https://doi.org/10.48550/arXiv.2302.13971>
- Morgan Ulinski. 2019. *Leveraging Text-to-Scene Generation for Language Elicitation and Documentation*. Ph.D. thesis, Columbia University, USA. <https://doi.org/10.7916/d8-2wmg-2230>.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008.
- Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. Blimp: The benchmark of linguistic minimal pairs for english. *Transactions of the Association for Computational Linguistics*, 8:377–392. https://doi.org/10.1162/tacl_a_00321

- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP*, pages 38–45. Association for Computational Linguistics. <https://doi.org/10.18653/v1/2020.emnlp-demos.6>
- Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. 2018. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, pages 1316–1324. IEEE Computer Society. <https://doi.org/10.1109/CVPR.2018.00143>
- Tianjun Zhang, Yi Zhang, Vibhav Vineet, Neel Joshi, and Xin Wang. 2023. Controllable text-to-image generation with GPT-4. *CoRR*, abs/2305.18583. <https://doi.org/10.48550/arXiv.2305.18583>
- Yian Zhang, Alex Warstadt, Xiaocheng Li, and Samuel R. Bowman. 2021. When do you need billions of words of pretraining data? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP*, pages 1112–1125. Association for Computational Linguistics. <https://doi.org/10.18653/v1/2021.acl-long.90>