Recurrent Instance Segmentation using Sequences of Referring expressions (Supplementary Material)

Alba Maria Hererra-Palacio
Universitat Politecnica de Catalunya

Carles Ventura
Universitat Oberta de Catalunya
cventuraroy@uoc.edu

Carina Silberer
Universitat Pompeu Fabra
carina.silberer@upf.edu

Ionut-Teodor Sorodoc
Universitat Pompeu Fabra
ionutteodor.sorodoc@upf.edu

Gemma Boleda
Universitat Pompeu Fabra
gemma.boleda@upf.edu

Xavier Giro-i-Nieto
Universitat Politecnica de Catalunya
xavier.giro@upc.edu

1 Related Work

The goal of visual grounding with referring expressions is to localize specific objects in an image by means of a natural language description of each object. Most existing approaches rely on scored bounding box proposals to determine the correct localization region with a bounding box. To obtain a more precise result, instance segmentation was proposed, which produces segmentation masks for the described objects.

This problem was firstly introduced in [3], where a CNN and an LSTM are used to extract visual and linguistic features, respectively. The fusion of the visual and linguistic representations was done by concatenating the LSTM output to the visual feature map at each spatial location. Then the low resolution output is up-sampled using deconvolution layers to yield the pixel-wise segmentation mask.

Instead of segmenting the image based only on a phrase embedding (i.e., referring expression or parts thereof), [7] exploits word-to-image interactions, directly combining visual features with each word feature, obtained from a language LSTM, to recurrently refine the segmentation results. [8] employs in a sequential manner both, a concatenation strategy to merge linguistic and visual features, and the computation of a dynamic filter, whose response is directly related to the presence of the object at a given spatial coordinate in the image. In [5], a convolutional LSTM (ConvLSTM) [11] model encodes and fuses visual and linguistic representations and outputs a rough localization of the referent. A recurrent refinement module then uses the fused representation and pyramidal image features to refine the segmentation by adaptively selecting and fusing image features at different scales.

To adaptively focus on informative words in the referring expression and important regions in the input image, cross-modal self-attention (CMSA) module [12] captures the long-range dependencies between linguistic and visual contexts to produce multimodal features. Moreover, a gated multi-level fusion module selectively integrates multi-level self-attentive features, corresponding to different levels in the image, which effectively capture fine details for precise segmentation masks.

In MAttNet [13], a language attention network decomposes referring expressions into three components, one for each visual module (subject, location, and relationship modules), and maps each of

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

them to single phrase embeddings. Given candidate objects by an off-the-shelf instance segmentation model and a referring expression, the visual module dynamically weights scores from all three modules to output overall scores. Two types of attention are used: (i) language-based attention, which learns the module weights as well as the word or phrase attention that each module should focus on, and (ii) visual attention, which allows the subject and relationship modules to focus on relevant image components.

2 Methodology

2.1 Referring expression encoder

In the following subsections it is detailed how contextualized phrase embeddings, inputs for the language encoder, are obtained from them.

The implementation of BERT embeddings used in our work[1] actually offers different off-the-shelf model variations. The ones available to extract embeddings from English text are:

- **Base model:** 12 encoder layers (transformer blocks), 768 hidden units and 12 attention heads.
- **Large model:** 24 encoder layers (transformer blocks), 1024 hidden units and 16 attention heads.

We use the `bert-base-cased model`, which corresponds to the base model and ignores casing.

A tokenizer is provided to pre-process the raw text data, since BERT is a pre-trained model that expects input data in a specific format. The tokenization steps for sentences are:

- **Text normalization:** Convert all whitespace characters to spaces. Additionally, lowercase the input and strip off accent markers for the uncased model.
- **Punctuation splitting:** Split all punctuation characters[2] by adding a space around them.
- **WordPiece tokenization:** Apply space tokenization to the output of the above procedure. Then tokenize each word separately. The tokenizer first checks if the whole word is in the vocabulary. If it is not, it tries to break the word into the largest possible subword tokens contained in the vocabulary, that will retain some of the contextual meaning of the original word. As a last resort it will decompose the word into individual characters. Note that because of this, we can always represent a word as, at the very least, the collection of its individual characters.

[1] `https://github.com/huggingface/pytorch-transformers`

[2] Punctuation characters are defined as any ASCII character different from a letter, number o space, or with a P* Unicode class.

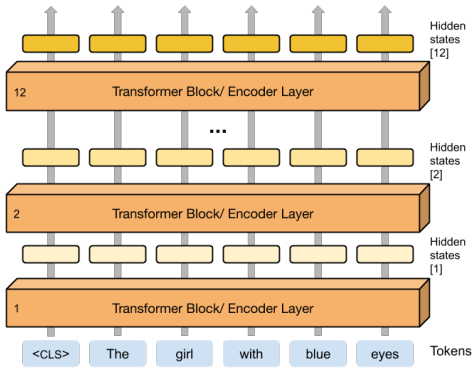


Figure 1: General scheme of BERT outputs used to generate contextualized embeddings. The output of each layer corresponding to a token can be used as a feature representing that token.

2.1.1 Pooled output

The pooled output encodes a whole sequence as a single vector. Every sequence always starts with the special classification token *[CLS]*, see Figure 1, which stands for classification. The last the state of the last hidden layer that encodes the *[CLS]* token is used as the aggregate sequence representation for classification tasks. The pooled output corresponds to that hidden state that is trained on the task of predicting the next sentence. As BERT authors remark, this output is usually not a good summary of the semantic content of the input.

2.1.2 Encoded layers

At the output of the last layer of the model, a set of contextualized embeddings (hidden states for this model) is stored in an four-dimensional tensor that has four dimensions, in the following order: layer number (12 layers/hidden states), batch number (1 sentence), word/token number in each sentence, and hidden unit/feature number (768 features). There are multiple strategies to obtain phrase contextual embeddings from them, which involve: (i) word embedding combination strategy, such as averaging, concatenating, pooling, etc., and (ii) sequence of hidden-states/layers used, such as last four, all, last layer, etc. In our work, we average the last hidden layer of each token producing a single 768 length vector for each referring expression.

2.1.3 Dimensionality reduction

A single embedding for each of our referring expressions is provided by the language encoder. To avoid memory problems while training the model and balance the dimension of the language and visual embeddings, we tried reducing the dimensionality of the textual embeddings. In our experiments we explore two ways of dimensionality reduction: principal component analysis (PCA) [9], and a learned linear projection, adding a dense layer that is trained with the model.

2.2 Image encoder

The still image is fed to the network as an RGB color image and encoded with ResNet-101 [2] model pre-trained on ImageNet [1]. The ResNet architecture is truncated at the last convolutional layer, thus removing the last two layers (pooling layer and classification layer). This architecture is finetuned for the specific task being solved, i.e. object segmentation by using referring expressions. The output of each convolutional block is used as an image feature, which provides a set of visual features at different resolutions, as shown in dark blue at the left of **??**. This visual encoding scheme was adopted from the former RSIS [10] model for semantic instance segmentation over still images.

2.3 Mask decoder

We explore several strategies to combine the language embeddings with multi-resolution visual features in the decoder. We wanted to preserve both linguistic and visual information in the segmentation, while avoiding high memory requirements during the training of the model. Therefore, to keep the inherent spatial information in the visual features when segmenting an instance, for each resolution, we concatenate the corresponding language embedding to each feature map along the channels' dimensions (depth) of the visual tensors. This allows every pixel embedding to receive the whole representation of the language information.

The recurrent architecture of the decoder, inherited from RSIS, allows to condition the predicted masks with those masks predicted from previous referring expressions over the same image. The ConvLSTM [11] layers used in the decoder contain a state memory that has the potential to remember previous predictions.

2.4 Loss

The detailed expression of the soft intersction over union used as loss function is

$$\text{sIoU}(m_1, m_2) = 1 - \frac{\sum_{i=1}^{M} m_{1,i} m_{2,i}}{\sum_{i=1}^{M} m_{1,i} + m_{2,i} - m_{1,i} m_{2,i}} \tag{1}$$

3 Experiments

3.1 Dataset

Our experiments use RefCOCO dataset [14], which provides both pixel-level masks and linguistic referring expressions for an image in complex real world scenes. The referring expressions for objects were collected in an interactive way with the ReferItGame proposed by [4]. Since each image contains multiple objects of the same category, the referrer must provide unambiguous relevant referring expressions, see an example in **??**.

RefCOCO contains $142,210$ referring expressions for $50,000$ referents (from 80 MSCOCO categories). These referents are depicted in $19,994$ images from the MSCOCO [6] dataset and the segmentation masks of the referents. We adopt the training, validation, and testA and testB splits provided by the University of North Carolina (UNC), which have respectively $42,404$, $3,811$, $1,975$ and $1,810$ referents. There is no overlap between train, validation and test images. Since half of the referents are people, people-vs-objects splits are used for testing, where testA images contain multiple people and testB multiple instances of all other objects.

3.2 Training details

The RefCOCO dataset provides several referring expressions for each segmented object. However, we only use one referring expression per referent in each epoch because we observed that using multiple ones was harmful for the performance of our model. This behaviour can be explained because of the spatial recurrence (memory) that RSIS contains. RSIS was actually designed for instance segmentation, in such a way that each pixel in the image can only belong to a single instance. This would be contradictory with using multiple referring expressions for the same referent.

Given a referent, referring expressions are chosen in the following way:

- During training, a referring expression is chosen randomly in each epoch. The training data consequently changes every epoch and acts as a sort of regularization.
- For validation and testing, we always use the first referring expression provided by the dataset, which speeds up validation and maintains consistency in the results. Even though not all referring expressions are used, embeddings obtained from referring expressions of the same referent are similar enough to obtain results representative of the whole dataset to compare with state-of-the-art methods. This hypothesis is validated comparing the results from testing with the first referring expression to the ones using a random one, which shows a minimal variation (on average less than 1% in overall IoU).

Given an input image, we resize it to $240x427$ pixels and keep the maximum number of referents to 6. The network is trained with ADAM optimizers with a weight decay of 10^{-7} and an initial learning rate of 10^{-4} (language encoder and decoder) and 10^{-7} (image encoder). We do not fine-tune the pre-trained BERT model during training. However, ResNet-101, which is used by the visual encoder, is fine-tuned with the architecture.

Experiments have been implemented with Pytorch in Python3.

3.3 Metrics

The quality of the predicted binary masks is evaluated with the overall and instance-level intersection-over-union (IoU) and $precision@X$. Overall IoU, referred as micro-average in NLP, calculates the total intersection area between predicted segmentation masks and ground truth divided by total union area accumulative over all the referents. A well-known issue regarding this IoU measure is its bias toward object instances that cover a large image area. To evaluate how well the individual instances

are segmented independently of their area, we evaluate the segmentation on an instance-level, i.e., the global IoU is obtained averaging across all referents, where for each referent IoU is calculated as the ratio between intersection and union of the prediction and the ground truth. The latter IoU measure, also known as macro-average in NLP, is the same used by the model to calculate the loss. The percentage of referents with an IoU score higher than a given threshold X is the $prec@X$ metric measure, set in the experiments to $X \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$.

3.4 Dimensionality reduction

We tested the impact of different ways to encode the referring expressions by the BERT-based language encoder, described in Section 2.1. The results on the RefCOCO dataset with a batch size of 32 images, randomly sorting the referents, are given in Table 1 showing the following comparisons: (i) BERT embedding: pooled output or encoded layers, (ii) transform for dimensionality reduction: None, using PCA or a trained linear projection, with a dimension of 768 or 128 or 64.

Even though both linguistic representations obtain similar results, encoded layers is slightly better. In general, results improve when applying a dimensionality reduction strategy. Overall, an embedding of dimension 64 works better. The best results are achieved with PCA. Probably the reason why the linear layer approach does not achieve better results is that it has too many parameters to learn. To see if the results could be improved, the linear layer can be initialized with the PCA values.

Instance IoU, which gives the same importance to each referent independently of their size, obtains higher results across all the configurations. Overall IoU, which does not take the size of the ground truth or the segmentation into account, obtains worse results since it is greatly affected by cases where the output mask selects large areas of the image.

Table 1: Results in RefCOCO for different embeddings configurations.

BERT	**Transform**	**Embedding dimension**	Instance IoU			Overall IoU		
			val	**testA**	**testB**	**val**	**testA**	**testB**
Pooled output	None	768	25.57	30.30	21.92	23.23	26.85	20.38
	PCA	128	27.70	32.17	24.27	25.00	28.25	22.82
		64	29.59	33.47	26.66	33.65	39.02	30.17
	Linear	128	26.44	30.88	23.50	23.76	27.38	21.67
		64	26.43	31.00	22.43	24.25	27.67	21.02
Encoded layers	None	768	26.86	31.32	23.04	24.51	27.85	21.58
	PCA	128	23.24	27.06	20.13	20.99	24.22	18.89
		64	**39.79**	**45.31**	**34.04**	**35.70**	**40.28**	**31.28**
	Linear	128	27.57	31.85	23.74	24.95	28.25	22.19
		64	36.36	42.27	31.94	32.23	37.06	28.60

3.5 Order of the referents and batch size

We study the effect of different configurations to train the model. Table 2 shows the impact of varying the batch size and the way referents are fed to the model. The latter can be sorted: (i) by area, (ii) randomly, each time an image is forwarded its objects are fed in a different order, which acts as a sort of data augmentation, and (iii) as in the RefCOCO dataset.

Sorting objects by area before feeding them to the model introduces a bias, learning to recurrently segment objects in the order of their area instead of fully exploiting the referring expressions for localization. Feeding the referents as in RefCOCO does not introduce such a strong bias, but since no data augmentation strategy is used, and the number of images in the dataset is limited, the model does not learn to generalize properly. From the results in Table 2, the best strategy is to randomly feed the referents, which acts as a sort of regularization which helps the model to generalize and reduces overfitting. For this configuration, smaller image batch sizes work better, a batch size of 16 images seems to be appropriate for this task and dataset.

Table 2: Results in RefCOCO with BERT encoded layers reduced with PCA to dimension 64 as linguistic embeddings.

Referent order	**Batch size**	Instance IoU			Overall IoU		
		val	**testA**	**testB**	**val**	**testA**	**testB**
Sorted by size	128	26.08	29.63	22.81	23.67	26.47	21.13
	32	26.12	28.66	23.82	23.88	25.81	22.23
Random	128	27.54	31.45	24.39	24.75	27.76	22.26
	64	33.17	37.42	30.51	29.52	32.77	27.03
	32	39.79	45.31	34.04	35.70	40.28	31.28
	16	**42.66**	**47.48**	**37.51**	**36.95**	**41.42**	**32.72**
RefCOCO	128	31.84	35.84	28.69	27.67	30.93	25.37

3.6 Qualitative results

Figure 2 shows some qualitative results generated by our network with examples on images from RefCOCO split testA, which is focus on multiple people. Similarly, Figure 3, shows some qualitative results for split testB. The depicted results are among the good predictions of the algorithm and show how our model can distinguish between different instances of the same class. Finally, in Figure 4 it can be seen that the order of the objects ids is consistent with the order of the referring expressions, showing that the predicted order has not been guessed by chance.

While there is still room for progress, these results indicate the potential of the proposed the task with the proposed deep neural architecture.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[3] Ronghang Hu, Marcus Rohrbach, and Trevor Darrell. Segmentation from natural language expressions. In *European Conference on Computer Vision*, pages 108–124. Springer, 2016.

[4] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 787–798, 2014.

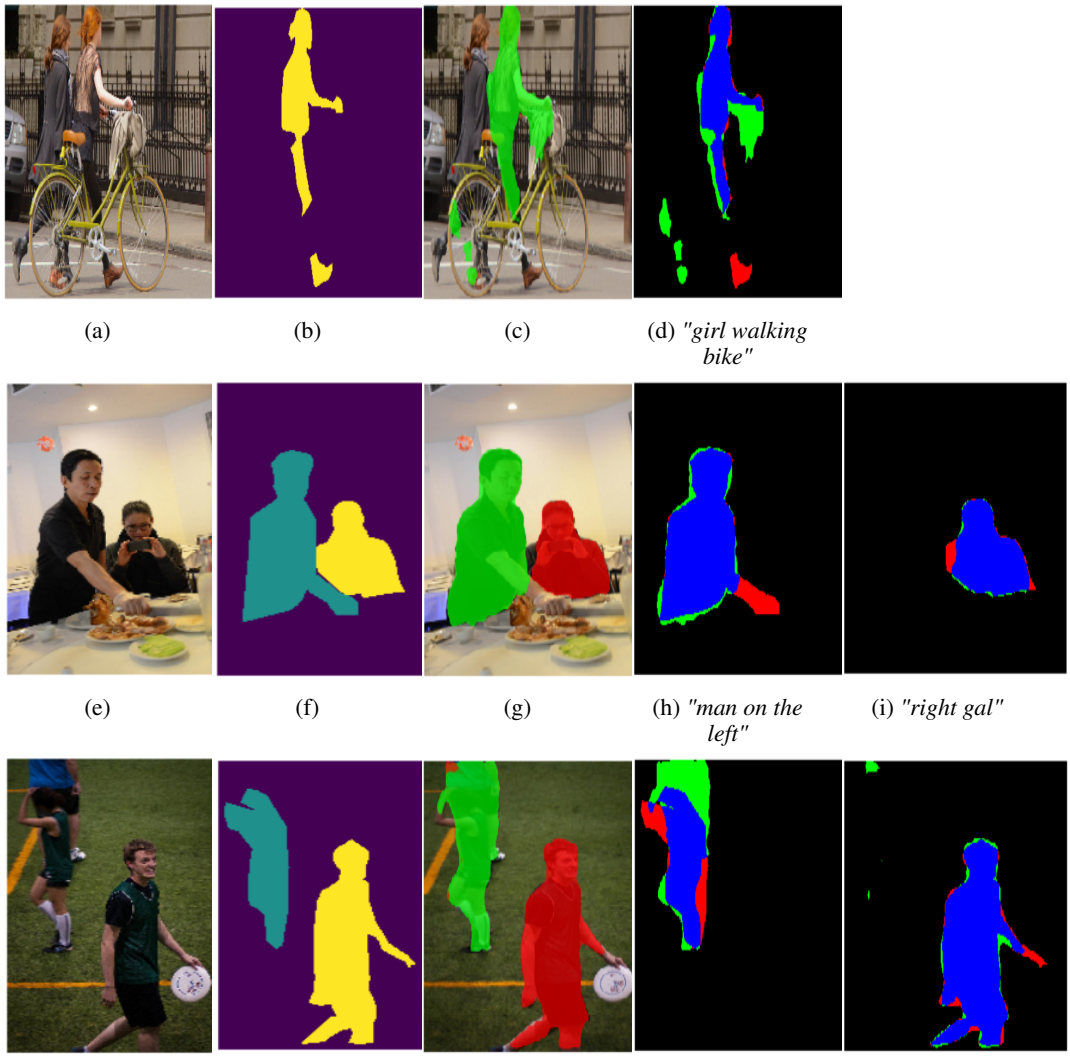
[5] Ruiyu Li, Kaican Li, Yi-Chun Kuo, Michelle Shu, Xiaojuan Qi, Xiaoyong Shen, and Jiaya Jia. Referring image segmentation via recurrent refinement networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2018.

[6] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[7] Chenxi Liu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, and Alan Yuille. Recurrent multimodal interaction for referring image segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1271–1280, 2017.

[8] Edgar Margffoy-Tuay, Juan C Pérez, Emilio Botero, and Pablo Arbeláez. Dynamic multimodal instance segmentation guided by natural language queries. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 630–645, 2018.

[9] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.



(a) (b) (c) (d) *"girl walking bike"*

(e) (f) (g) (h) *"man on the left"* (i) *"right gal"*

(j) (k) (l) (m) *"player on the left turned to the side"* (n) *"the dude"*

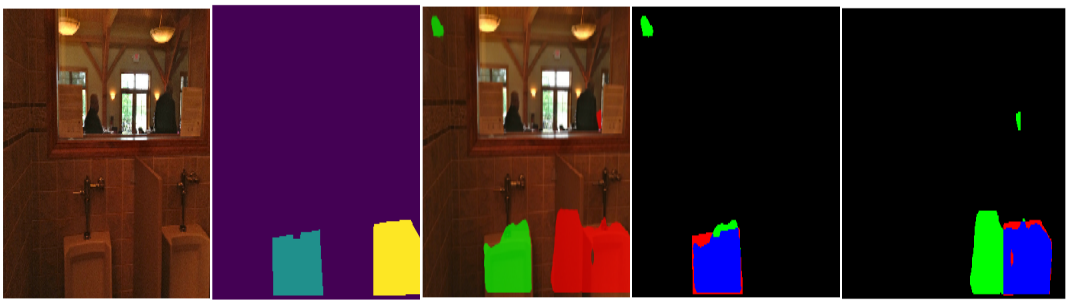
Figure 2: In RefCOCO testA, from left to right: original image, ground truth, segmentation result, visualization of the results, where red pixels are false negatives, blue true positives, green false positives and black true negatives.

[10] Amaia Salvador, Miriam Bellver, Victor Campos, Manel Baradad, Ferran Marques, Jordi Torres, and Xavier Giro-i Nieto. Recurrent neural networks for semantic instance segmentation. *arXiv preprint arXiv:1712.00617*, 2017.

[11] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015.

[12] Linwei Ye, Mrigank Rochan, Zhi Liu, and Yang Wang. Cross-modal self-attention network for referring image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10502–10511, 2019.

[13] Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L Berg. Mattnet: Modular attention network for referring expression comprehension. In *CVPR*, 2018.

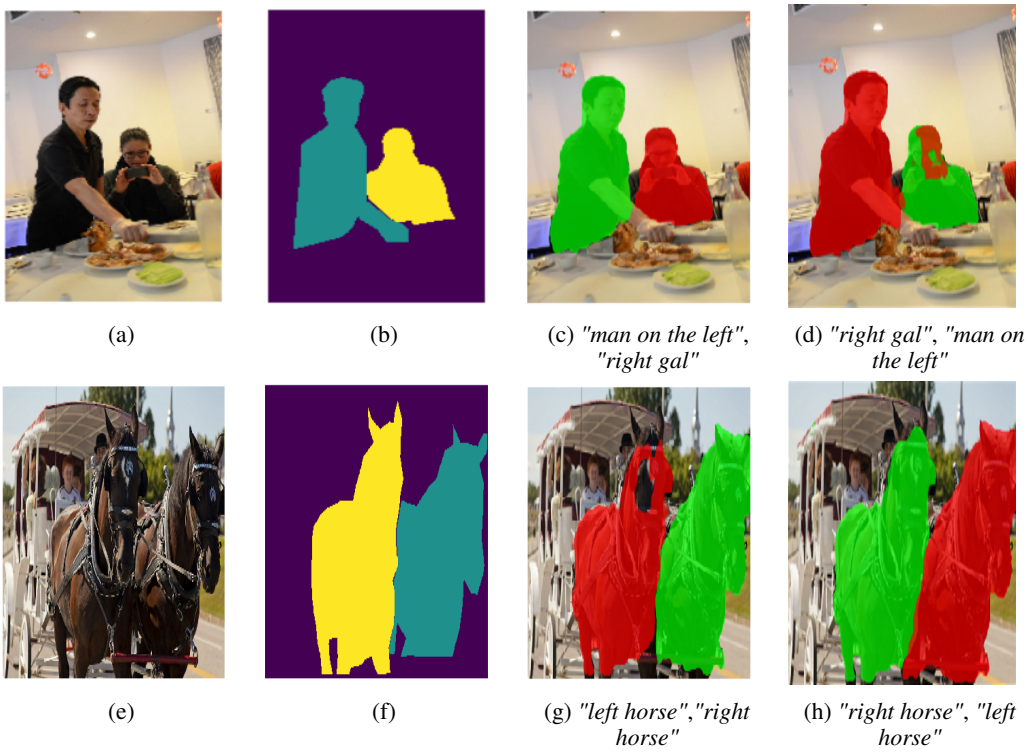


(a) (b) (c) (d) *"urinal on the left"* (e) *"right urinal"*

(f) (g) (h) (i) *"top zebra"* (j) *"red object"*

Figure 3: In RefCOCO testB, from left to right: original image, ground truth, segmentation result, visualization of the results, where red pixels are false negatives, blue true positives, green false positives and black true negatives.

[14] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. Modeling context in referring expressions. In *European Conference on Computer Vision*, pages 69–85. Springer, 2016.



(a) (b) (c) *"man on the left", "right gal"* (d) *"right gal", "man on the left"*

(e) (f) (g) *"left horse","right horse"* (h) *"right horse", "left horse"*

Figure 4: Examples of results obtained changing the order the referring expressions are fed to the model. From left to right: original image, ground truth, segmentation result (original order), and segmentation results (inverse order).