Artificial Intelligence 274 (2019) 44–65

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Distributional semantics of objects in visual scenes in comparison to text

Timo Lüddecke [a,*], Alejandro Agostini [a], Michael Fauth [a], Minija Tamosiunaite [a,b], Florentin Wörgötter [a]

[a] Georg-August-University Göttingen, Third Institute of Physics, Germany
[b] Vytautas Magnus University, Faculty of Informatics, Lithuania

ARTICLE INFO

Article history:
Received 24 July 2017
Received in revised form 31 May 2018
Accepted 4 December 2018
Available online 7 February 2019

Keywords:
Object semantics
Vision and language
Semantics
Distributional hypothesis
Computer vision

ABSTRACT

The *distributional hypothesis* states that the meaning of a concept is defined through the contexts it occurs in. In practice, often word co-occurrence and proximity are analyzed in text corpora for a given word to obtain a real-valued semantic word vector, which is taken to (at least partially) encode the meaning of this word. Here we transfer this idea from text to images, where pre-assigned labels of other objects or activations of convolutional neural networks serve as context. We propose a simple algorithm that extracts and processes object contexts from an image database and yields semantic vectors for objects. We show empirically that these representations exhibit on par performance with state-of-the-art distributional models over a set of conventional objects. For this we employ well-known word benchmarks in addition to a newly proposed object-centric benchmark.

© 2019 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. Introduction

It remains a matter of debate, which aspects constitute the *meaning* of a word or of an object in a scene and the term meaning is heavily discussed in different fields. Here we are specifically concerned with the distributional representation hypothesis by Harris [17], which states that the company of a word determines its meaning (*distributional semantics*). This study sets out to test this hypothesis on images.

For the definition of meaning in the above sense, natural language processing (NLP) uses semantic vectors, which represent word-neighborhoods in a sentence. This approach has proven to be useful in many different applications, e.g. for text translation between different languages [27], for determining the sentiment of a sentence [33] and for question answering [39]. Thesaurus generation, spelling correction, and query expansion count among further applications discussed by Turney and Pantel [37].

Analogously, context in visual scenes is also important for defining the meaning of objects. It might serve as a basis for a variety of approaches in image understanding that involve interactions across multiple objects, e.g. determining useful robotic actions or finding task-relevant objects in a scene. Thus, in this work, inspired by NLP approaches, we develop methods to obtain semantic vector representations of objects by considering their respective contexts in real world scenes that are composed of multiple objects.

* Corresponding author.
E-mail address: timo.lueddecke@phys.uni-goettingen.de (T. Lüddecke).

https://doi.org/10.1016/j.artint.2018.12.009
0004-3702/© 2019 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

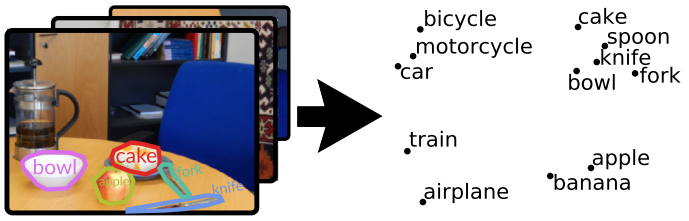

Fig. 1. By assessing object co-occurrences and visual features we obtain semantic vectors for objects (right, actual output).

Texts and scenes are akin structures, both being composed of many individual but inter-related constituents: words or objects. The location of a word in a sentence but also that of an object in a scene is subject to some constraints. On the one hand, these constraints can be *fundamental*, e.g. imposed by grammar or physics. The violation of these constrains will render a sentence wrong or make a scene impossible or nonsensical. E.g., grammar forbids two subsequent articles as much as a chair can not stand on the ceiling due to gravity.

On the other hand, obeying only grammatical constraints does not guarantee that a sentence will make sense and the laws of physics will also not guarantee that a room has a useful structure. For making sense also the respective neighborhoods need to be appropriate, i.e. the context in which words are used or the arrangement of items in a room.

Furthermore, context can help to disentangle multiple meanings of words or objects. For example in natural language, the meaning of a polyseme depends on its context and, similarly, multi-functional objects are employed differently depending on the situation. In the same way that e.g. the word “board” acquires a particular sense (out of many) by contextual words, objects surrounding a coffee cup in a scene constrain the set of useful actions involving the cup (e.g. at a coffee klatsch v.s. when the same cup is in the sink with dirty dishes).

However, it is only text analysis where there has been a long history of approaches that address the question of distributional semantics and that derive the meaning of a word, at least to some degree, from context. Interest in these approaches recently revived by the success of large-scale methods [28,29] exhibiting remarkable performance in judging similarity or analogy of concepts. By contrast, little work has been done on obtaining meanings of objects by considering their scene contexts. This should, however, be possible, in particular due to the large-scale data-sets that have recently been published in the computer vision community, which allow now for the investigation of distributed semantics in the domain of objects. Therefore, it seems justified to investigate the learning of semantic vectors not only of words but also of objects.

In this work we study the hypothesis that the spatial context contributes to the meaning of an object in a scene, analogously to surrounding words defining the meaning of a word. To gather evidence we design an algorithm that extracts semantic vectors from scenes as visualized in Fig. 1. We aim at analyzing a big enough set of images to arrive at a representation, where semantic vectors for similar objects would group together. Fig. 1 schematically shows this for *cake*, *spoon*, *fork* and *knife*, which group together (see schema with object names) but remain separate from *bicycle*, *motorcycle* and *car*, which form a different group. Just from common sense one would hope to obtain such clustering results, but there are obvious differences between sentences and scenes. In contrast to a scene, a text is a linear structure, i.e. each word has a predecessor and a successor. Thus, while there is the straightforward assumption that the distributional hypothesis should also hold for images, it remains quite a question whether or not the more complex 3D layout of the visual world (or its 2D image projections) might not render context relations too spread out? Hence, we ask: Will scenes provide equally strong context relations than text? In this paper we address this question comparing a large set of different NLP as well as image-based methods.

1.1. Overview of the approach

A schematic introduction to our approach in comparison to linguistic approaches is presented in Fig. 2. Common distributional models (top) take natural language text corpora, determine word sequences, and generate context vectors by word co-occurrence. A context vector describes the surrounding of individual entities (such as words) by an array of real numbers. Different methods are used to combine such vectors so that finally semantic vector representations emerge for different concepts that can be compared. In essence, our approach (middle) is similar, but it only considers concepts that are objects. We take scene datasets and extract different image descriptors, which are (see numbering in the figure): 1) Human-assigned object labels, 2) object labels automatically obtained by applying an RCNN [14] to detect objects, and 3) CNN activations, which are features generated using pre-trained convolutional neural networks. From all three approaches we extract context vectors for each object in each scene. Context vectors are then merged together to create the unique semantic vector for a specific object class. This allows directly comparing not only the three types of descriptors but also benchmarking our results against automatic NLP-based methods (top) as well as against human rater-based methods (bottom). This is done by measuring semantic distances between objects (the inverse of semantic similarity) in the respective semantic vector spaces.

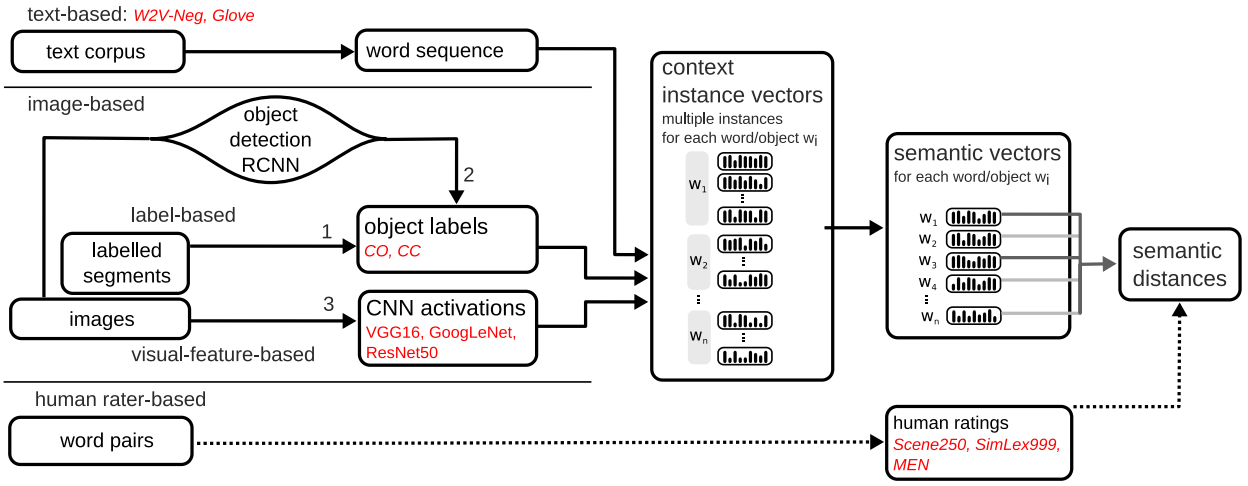

Fig. 2. Flow diagrams of the analyzes performed here with standard natural language processing, our new approach, as well as based on human labeling. In red, we show the abbreviation of the different analysis methods used (for descriptions see Methods). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

1.2. Contributions

We propose a simple algorithm to compute semantic vectors for object classes in segmented images and extensively compare this with existing (text-based) methods. This paper is to our knowledge the first to specifically focus on objects in scenes. In addition to the analysis of existing data, we also collected a dataset of similarity and relatedness judgments for 250 object pairs from 20 raters specifically slanted towards everyday objects.

Our analysis shows that the obtained image-based representations are on par with existing text-based representation methods. Quality can be further improved by concatenating different context models. These findings indicate that not only text might serve as a basis for distributional semantics but also visual scenes, which are fundamentally different from text.

The remainder of this paper is structured in the following way. Literature, mostly originating in the natural language processing community, is reviewed in Section 2. Section 3 introduces the theoretical background of our approach and section 4 the experimental methods. In order to assess the validity of the approach, experiments investigating the semantic similarity are carried out in Section 5. Discussion and outlook are provided in Section 6.

2. Related work

There are many approaches originating from diverse fields that can be related to our work. They are discussed in this section and ordered according to the field they come from. Most similar to the idea of this paper are approaches that link distributed semantics from natural language processing with images. However, the field of natural language processing arguably has the largest impact on this work and is consequently discussed next in more detail. Image labeling is discussed, too, as it links visual with textual data.

Natural language processing. The idea of representing entities through distributional representations has its origins in linguistics. The distributional hypothesis introduced by Harris [17] established the field of distributional semantics. The idea in distributional semantics is to statistically analyze the distribution of words or other linguistic entities in order to derive a meaning or simply put: “You shall know a word by the company it keeps.” [12,37].

Several methods represent contextual concepts as real numbered vectors, either as an intermediate representation to be used in various NLP tasks [4,8,2] or, as we do in this work, as the final output to be assessed with respect to semantic similarity [36].

An elaborate survey has been conducted by Turney and Pantel [37]. The work by Lund and Burgess [26] is particularly relevant as their co-occurrence-based method is closely related to our approach but differs with respect to the derivation as well as the data underlying the methods. A direction of research, which is only laterally addressed by Turney and Pantel [37], is based on parameterizing more complex statistical models by optimizing from random initializations. Bengio et al. [4] proposed to condition the probability of a word on the context of co-occurring words by a neural architecture. This system learns to represent words as vectors and to employ these representations in order to infer the word probability. Recently, starting with the *word2vec* algorithms [28], large scale models have gained a lot of attention due to their ability to extract synonyms and analogies with remarkable quality for numerous words in an unsupervised way from text collections. Subsequent papers discussed further ways of obtaining word vectors exhibiting these properties [29] and related them to traditional approaches [23].

Image labeling. Vector representations are not only employed in natural language processing but also in computer vision. Some image labeling methods learn to map images and labels (i.e. text) into a joint space. One of these approaches is the WARP model of Weston et al. [38]. Images are mapped to this space by multiplying a bag-of-visual-word representation of the image, which can be obtained by quantization of features [9]. Learning is carried out by an optimization procedure starting from random initializations for the mappings and then minimizing the hinge rank loss, given ground truth data. Having learned the mappings, unknown images can be mapped into the vector space and, by determining the nearest neighbors, labels can be predicted.

Instead of training from scratch, a more recent approach [13] makes use of transfer learning. They learn a mapping for images into the word vector space obtained using the skip-gram model proposed by Mikolov et al. [28]. The mapping is carried out by a pre-trained convolutional neural network that is fine-tuned to adapt to the task using a hinge rank loss. At test time, the image is mapped into the joint vector space and its nearest neighbors are evaluated in order to predict a category.

Both approaches differ from our methods because they aim at attributing labels to images based on their visual content rather than obtaining general vector representations of objects. They focus on images presenting a single salient object (which is common in ImageNet) whereas our work is interested in constellations of multiple objects.

Distributed semantics and images. Linking distributed semantics from natural language processing with images is not a new idea. There has been some work employing labels or tags of images in order to link textual meaning with visual features extracted from the images.

Feng and Lapata [11] propose a method that works on documents with an associated image and experiments are conducted on news articles. For both modalities, a bag-of-word representation is used with the image's representation being obtained through extracting visual words. SIFT descriptors [25], which capture local edge orientations, are extracted from images and matched to visual words, which have been learned previously by k-means clustering of all feature descriptors from all images. In this way, visual words can be treated like actual words in a text, i.e. semantic vectors are assigned to words. A topic model, based on Latent Dirichlet Allocation [5], is trained on documents involving both, textual and visual words. In their experiments, the model is used to measure similarity between words.

Bruni et al. [6] also employ visual words descriptions of images. However, visual features are computed differently: SIFT is extracted densely (without keypoints) with spatial binning from each channel of a HSV encoded image, making the algorithm sensitive to color. Also, a simple count model is employed rather than a topic model. Visual words are used in conjunction with multiple textual labels (tags) to obtain co-occurrence counts for certain terms. Finally, they are transformed into Local Mutual Information association scores. Multiple models of fusing representations from the modalities are discussed.

Kádár et al. [22] propose a model to learn meanings of words using images and their caption text, relating words to visual features. Similarly to our work, they make use of pre-trained Deep Convolutional Neural Networks (DCNNs) to extract visual features. A similar approach is pursued by Kiela and Bottou [21]. Rather than relying on captions, they concatenate visual features of multiple object classes of ImageNet [10], which have been extracted using pre-trained DCNNs with the well-known word semantic vectors from Mikolov et al. [28]. Both assess their models on common word similarity benchmarks. Another example of a DCNN-based approach is by Peterson et al. [30]. They extract features from 120 photographs of animals and observe that the pairwise dot similarity between the features only correlate weakly with human similarity judgments. Consequently, a transformation of the visual features is learned which strongly increases the correlation with human judgments.

All the discussed approaches differ from our work in the aspect that labels or text refer to the entire image rather than only to regions of it. In our setting, due to availability of segmentations, we are able to directly access both, object location in the image and class of the object, which enables us to explicitly focus on the context of objects rather than the features of the object itself and with this we can evaluate the validity of the distributional hypothesis in images.

3. Determining image-based context

One of the central assumptions of the approaches discussed above is that context strongly determines meaning. In this section we will, therefore, first describe how we define object context in an image and then how we integrate contexts extracted from separate images to obtain context representations over an image database. We call context obtained for one object in one image *context instance* or *context instance vector* and the context representation of an object integrated over many instances we call *semantic vector*. As our main approach for integration is context averaging, we will also provide a mathematical justification for that. For comparison, we also use the standard skip-gram method and the median for integration. Finally, we explain how we concatenate different features to create fused context representations.

3.1. Context instance definition

Given a dataset of scenes containing multiple objects, context extraction is applied on each object in each scene to obtain the set of context instances for each object. In this study we used two main approaches to define object instance context in an image: *label-based* context, described in subsection 3.1.1 and *visual feature-based* context, described in subsection 3.1.2.

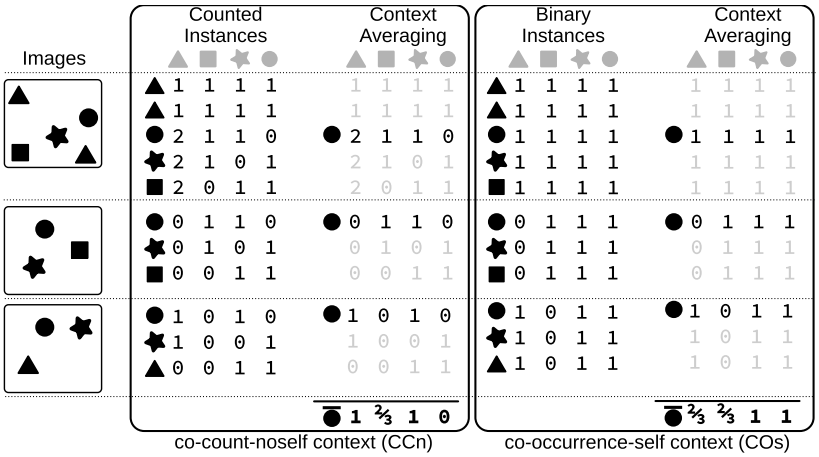

Fig. 3. The context averaging method considers each object (illustrated by shapes) in each image and determines its context. Subsequently, the averaged context is calculated across all objects and used as a semantic vector. Contextual instances can either be counted (co-count) or just indexed (co-occurrence) and the reference object can be counted among the context (self) or not (noself).

Table 1
Comparison of the employed pre-trained CNNs. The layer from which features are extracted, the number of features as well as top-5 accuracy achieved in the ImageNet classification challenge are provided.

Network architecture	Feature Vector Layer	Feature Vector Size	reported top5-accuracy
InceptionV1	AvgPool_1a_7x7	1,024	89.6 %
InceptionV4	global_pool	1,536	95.2 %
ResNet50	global_pool	4,096	89.8 %
VGG16	fc7	2,048	92.8 %

3.1.1. Label-based

Label-based contexts fully rely on image annotations, i.e. image pixels are not taken into consideration. An illustration of a context description for this case is provided in Fig. 3. Three schematic images are shown (left), where objects are represented by abstract shapes. Two ways of determining context are here defined: For each object in the scene all neighboring objects for each of the other object classes are counted. The resulting counts are used as the components of a vector, which is then taken as the instantiation of this object's context. We call this model *co-count* (CC, see column called "Counted Instances" in the left box in Fig. 3). Alternatively, we can neglect the number of objects in the scene and just tick-mark every existing object creating a binary (there vs. not-there) vector. This model will be referred to as *co-occurrence* (CO, see column called "Binary Instances" in the right box in Fig. 3). The reference object (the object we currently consider) can be counted among the context or not, which we call *s* or *n* (*self* or *noself*, Fig. 3, right vs. left). Hence for a label-based vector v holds: $v \in \mathbb{N}^N$ with N being the number of object classes in the dataset, in case of co-occurrence even $v \in \{0,1\}^N$.

3.1.2. Visual feature-based

An alternative, yet also image-based, way of describing object context is through visual features. While multiple approaches to extract visual features from an image are possible, e.g. bag-of-visual words [9], recent years have been dominated by the success of convolutional neural networks (CNNs), where activations in specific layers can be considered as an abstract representation ("features") of the image that was fed to the CNN. Here we constrained our analysis to four CNNs which were trained on ImageNet [10]: *InceptionV1 (GoogLeNet)* [34], *InceptionV4* [35], *VGG16* [32] and *ResNet50* [18]. In contrast to label-based vectors, visual feature-based vectors are more general. If o is an object with an associated image then the visual feature-based context $\mathbf{f}$ of o is defined as the result of the sequence of the interleaved matrix multiplications, non-linearities, and pooling operations performed in the CNN up to the layer from which $\mathbf{f}$ is extracted. Visual-feature-based context is described as a vector of real numbers as opposed to a vector of natural numbers for the label-based context, above. Technical details of the employed CNNs are shown in Table 1, where *Feature Size* indicates the number of features we were actually using.

Following the distributional hypothesis [17], we constrained our analysis to the image region that pertains to the context of the object but exclude the object itself. In the visual feature-based case this means that we extract CNN features after covering the reference object with black box mask, such that not even the shape of the object is visible. In case there are multiple objects considered all this is done for every one of them.

From a theoretical perspective, we expect a black box in the image to inhibit the features of the masked area and this way emphasizing the features extracted from the unmasked area – the context. If the convolution window contains both, mask and contents, there might be inference. However, this is a bias that affects the "context" of every object and even if

"wrong" features are now introduced, they are canceled out when analyzing the differences (distances) between two object representations. Actually, we experimented with an object-shaped (instead of rectangular) black mask, which yielded similar results.

3.2. Context integration

Context integration is used to obtain the semantic vectors. Three methods are compared: context averaging, skip gram based context integration, and the use of the median instead of the average.

3.2.1. Context averaging

One way to obtain the semantic vector of an object is to calculate the element-wise average of the context vectors obtained for individual object instances. This seems overly simple but it is in fact the maximum likelihood estimator of the underlying distribution(s) of objects in images, as shown below.

Let the set $\mathcal{O}(k)$ denote all object instances of class k in the dataset. $\mathbf{f}$ is any context extraction function that considers an object's context and returns a real-valued vector representing this context. Examples of such functions are described in subsection 3.1.1 and 3.1.2. The vector representation $\hat{\mathbf{\Lambda}}_k$ for concept k is then obtained by this formula:

$$\hat{\mathbf{\Lambda}}_k = \frac{1}{|\mathcal{O}(k)|} \sum_{o \in \mathcal{O}(k)} f(o) \tag{1}$$

The averaging method is illustrated in Fig. 3 for the "black-disk-object" $\bullet$. Each highlighted row corresponds to the context vector of the black disk for the corresponding schematic image on the left side, hence to a vector $\mathbf{f}(o)$ in equation 1. In the depicted case $k = \bullet$ the semantic vector $\hat{\mathbf{\Lambda}}_\bullet$ is obtained by averaging the individual context vectors.

For obtaining semantic vectors when using visual-feature-based analysis the real-numbered context instance vectors are averaged in the same way to obtain object class representations.

*Justification for context averaging.*The averaging of context can be justified by imagining a scenario where the scene is scanned from the perspective of an object o and considering each time another object is seen as an event, which is dependent on the object class of o. In the co-count model (CC), the number of such events (per object class) is described by integer random variables, while in the co-occurrence model (CO) the random variables refer to the probability of the event. Consequently, the underlying distributions are for co-count *Poisson* and for co-occurrence *Bernoulli*. For both distributions, averaging all observations is equivalent to the maximum likelihood estimation of the distribution's parameters. Hence, we implicitly define the meaning of objects to be optimal parameters of random distributions over visual features or object labels of given observations in a large scale dataset.

The theoretical motivation for averaging rests on the following arguments first discussed for the co-count model. As stated above, scanning a scene under the co-count model implies that the number of contextual objects of type i given the object class k is Poisson distributed, i.e. $\mathbf{c}_i \mid k \sim \mathit{Poisson}(\mathbf{\Lambda}_{ki})$ with each $\mathbf{c}_i \in \mathbb{N}_0$. Hence, the corresponding likelihood function is

$$\mathcal{L}(\lambda_{ki} \mid \tilde{c}_{ki}^{(0)}, \ldots, \tilde{c}_{ki}^{(n)}) = \prod_{j=1}^{N} \frac{\lambda^{\tilde{c}_{ki}^{(j)}} e^{-\lambda}}{\tilde{c}_{ki}^{(j)}!}$$

with $\{\tilde{c}_{ki}^{(j)} \mid j = 1, \ldots, N_k\}$ being the observed counts of object i in the context of object class k. Then the maximum likelihood estimator for each λ_{ki} is the sample mean.

An alternative model is the co-occurrence model. Here, we only mark object existence. Hence, we assume that encountering one or more objects of class i in the context of class k has a certain success probability. This leads to $P(\mathbf{c}_i \mid o = k) \sim \mathit{Bernoulli}(p_{ik})$ with each $\mathbf{c}_i \in \{0, 1\}$ and the likelihood function

$$\mathcal{L}(p_{ki} \mid \tilde{c}_{ki}^{(0)}, \ldots, \tilde{c}_{ki}^{(n)}) = \prod_{j=1}^{N} p^{\tilde{c}_{ki}^{(j)}} (1-p)^{(1-\tilde{c}_{ki}^{(j)})}$$

Also here a maximum likelihood estimation of the parameters p_{ki} is represented by the sample mean. In the Appendix we provide additional derivations concerning these aspects.

3.2.2. Skip-gram-based context integration

In addition to the counting-based averaging described above we implemented a skip-gram-based context integration method, which originates from natural language processing and is considered to be state-of-the-art therein. Its goal is to generate representations of words that, given one word from a corpus, allow for the prediction of contextual (surrounding) words. This algorithm serves as the basis for the word2vec [28] programs, which have been shown to yield very good semantic word representations.

We implemented the skip-gram negative sampling algorithm [28], which approximates the skip-gram target in a computationally efficient way, by using a fixed number of negative-sample-pairs. This method is only compatible with co-occurrence (CO) context. In our case, positive samples are obtained from pairs of objects within an image. Training is carried out for 50 epochs with a learning rate of 1. For every positive sample, 25 negative samples are presented and the dimension of the concept vectors is 64. A detailed exploration of hyperparameters leading to values reported above is provided in section A.6 in the Appendix.

3.2.3. Median-based context integration

In general, there would be many more integration methods possible and we had to limit ourselves to some of the most common ones, where using the median appears another possible "natural" choice. Results are not impressive and therefore we show this only in the Appendix to not overburden the main text.

3.3. Fusion of contexts

Label-based and visual feature-based contexts encode different information. The former tells us something about co-occurring object classes using a high-level (symbolic) description, while the latter addresses visual appearance (features) of the context-objects. Hence, it appears natural to combine both and see whether the semantic quality improves. While there exist various possible methods, we will emphasize on concatenation. The advantage of this method is that it does not require strong assumptions on the structure of the context vectors, i.e. it can be used to combine any number of real-valued vectors, also in case they have different length.

Concatenation can be applied at two levels: Given a scene and one object in that scene, we can apply several context extraction methods, e.g. label-based context and visual-feature-based context and concatenate both context vectors. This procedure we call *early fusion*. Alternatively, we can integrate contexts into semantic vectors using the two methods independently and then concatenate the obtained vectors; e.g., we first average all visual feature-based contexts of "car" and average all label-based contexts of "car" and then concatenate the resulting two vectors. This we call *late fusion*.

Let $\mathbf{f}$ and $\mathbf{g}$ be two context extraction methods, $O(k)$ the set of all object instances of class k in the dataset, and ϕ denote the context integration function (e.g. averaging).

Then we can formally define early fusion as

$$\Lambda_k^{early} = \phi(\{\mathbf{f}(o) \,||\, \mathbf{g}(o) \mid o \in O(k)\})$$

with $||$ being the concatenation operation which involves normalization using mean and standard deviation of each vector:

$$\mathbf{a} \,||\, \mathbf{b} = \left(\frac{\mathbf{a} - \mu_{\mathbf{a}}}{\sigma_{\mathbf{a}}}, \frac{\mathbf{b} - \mu_{\mathbf{b}}}{\sigma_{\mathbf{b}}}\right).$$

Late fusion is defined by:

$$\Lambda_k^{late} = \phi(\{\mathbf{f}(o) \mid o \in O(k)\}) \,|||\, \phi(\{\mathbf{g}(o) \mid o \in O(k)\}),$$

with $|||$ also being a concatenation operation but here normalization is made in a different way. Let $\mathbf{a}$ be the semantic vectors obtained by the first of the two to-be-fused context extraction methods for each object included in the study. Then we arrange those semantic vectors into matrix A row-wise (each vector is one row). Analogously, into matrix B semantic vectors obtained using the second context extraction method are arranged. From those matrices we obtain the vectors of the means: $\mu(A)$ and $\mu(B)$, and their standard deviations: $\sigma(A)$ and $\sigma(B)$ now column-wise, so that each semantic vector is zero-centered and normalized. Then the concatenation operation uses this normalization:

$$\mathbf{a} \,|||\, \mathbf{b} = \left(\frac{\mathbf{a} - \mu(A)}{\sigma(A)}, \frac{\mathbf{b} - \mu(B)}{\sigma(B)}\right)$$

where subtraction and division operations are performed element-wise.

In section A.2 in the Appendix we introduce and analyze more fusion techniques.

4. Standard methods from NLP

One major goal of this study is to compare our new image-based methods against state-of-the-art text-based methods for the extraction of semantic vectors from NLP (see top in Fig. 2). For this we used Glove [29] (50 dimensions, 6 billion tokens) and the skip-gram negative-sampling method (part of Word2Vec toolkit, 300 dimensions, 100 billion words, Google News dataset) [28]. In both cases, we downloaded pre-computed semantic vectors provided by the authors from the Internet. As these are standard methods, we do not extend explanations here and instead refer the reader to the respective references [28,29].

Table 2
Overview of image datasets used to obtain vector representations. LabelMe dataset abbreviated as LM.

Image Training datasets			
Dataset	#Classes	#Scenes	#Instances
COCO	80	82,081	604,907
LM	3,288	28,072	325,288
RCNN	79	111,974	286,835

Text Training datasets		
Method	Dataset	#Words
W2V-Neg	GoogleNews	100B
Glove	Wikipedia + Gigaword 5	6B

Human-rater datasets		
Name	#Words	#Pairs
Scene250	80	250
Simlex999	1028	999
MEN	751	3000

5. Datasets

5.1. Image datasets

Since our method requires annotated image data, we started our experiments using datasets offering object bounding boxes as well as object labels for each box. However, to show that our method can also be used in case where only raw images are available, we took an additional dataset that is initially not annotated and we used an advanced off-the-shelf object detection-and-labeling algorithm to perform automatic labeling. Specifically, we used COCO [24] and LabelMe [31] as annotated source datasets. We created the additional, automatically labeled dataset, which we will call RCNN through-out this paper, from a combination of images from the COCO and ImageNet [10] datasets. Dataset statistics are presented in Table 2.

5.1.1. Manually annotated datasets

Both, COCO and LabelMe provide annotations in form of object segments with corresponding object names. The COCO dataset has 7.4 object labels per scene on average and the LabelMe has 11.6 object labels. The object labels in COCO are very reliable, but the dataset contains only 80 object classes, while LabelMe provides a more diverse set of classes but has a tendency to contain wrong or nonsensical labels due to being crowd-sourced. Because of this, we only consider object labels that occur at least three times in LabelMe, which yields 3288 object classes.

5.1.2. Automatically annotated dataset, RCNN

We have created the RCNN dataset by applying the *Mask R-CNN* object detector [19] on unannotated images. Mask R-CNN takes a raw image as input and automatically generates labeled bounding boxes for objects detected in the scene. This way an annotated dataset is created, which is analogous to the manually labeled datasets introduced in the subsection 5.1.1 above. We used on purpose the same initial dataset, namely COCO, to create RCNN, to allow for a direct comparison with the results from the human-labeled data. We were, however, forced to use the smaller *testing set* of COCO, because the pre-trained Mask-R-CNN model we employ was trained on the COCO-training set. Thus, to increase the resulting RCNN dataset, we added a subset of ImageNet images. With this, we processed 160,000 images in total and Mask R-CNN found objects in 111,974 images therein, because of a high threshold we applied for reliable object detection. By this procedure, 79 object classes were discovered from the possible 80 COCO objects classes for which the Mask R-CNN was pre-trained. The average number of discovered objects per scene is 2.5.

5.2. Human rater-based datasets (Ground Truth Benchmarks)

To compare our scene-based analysis with the corresponding results obtained in natural language analysis, we conducted extensive experiments on common human rater-based benchmarks from NLP including a newly introduced benchmark specifically targeting everyday objects.

All these benchmarks rely on word-pair comparisons assessed by human raters. In general, two relations are measured: *similarity* and *relatedness* [20]. *Similarity* refers to shared properties between the two words (concepts) in a pair; e.g. a train and a car both have wheels, move on their own and host passengers. On the other hand, the more general aspect *relatedness* incorporates additional relationships [7], like function or meronymy; e.g. a remote control is used to turn on the TV, hence, both are related but not very similar.

5.2.1. Existing word pair-based benchmarks

Here we used the MEN ([6], 3000 word pairs) and SimLex999 ([20], 999 word pairs) benchmarks. Naturally, we had to restrict all analyses to the intersection of words between benchmark- and image datasets, which is rather small (for quantification of intersection see section A.3 in the appendix).

5.2.2. Novel object name-based benchmark

Due to the small intersection found for the common benchmarks, we decided to create our own benchmark dataset, called *Scene250*, which consists of a randomly sampled subset of 250 pairs from the possible 3160 pairs made by the 80 COCO objects. This way, *Scene250* fully intersects with COCO and also to a large degree with LabelMe (see section A.3). We asked 20 adult raters (10 male, 10 female) to indicate the degree of object similarity and relatedness in those pairs using a scale from zero to ten. The concepts of similarity and relatedness were explained to the participants based on definitions and examples. For obtaining the final score we averaged the scores from all 20 participants. The average pair-wise Spearman correlation coefficient across raters was 0.544 for similarity and 0.659 for relatedness.

The *Scene250* dataset is available for download at: http://cns.physik3.uni-goettingen.de/cns-group/datasets/scene250/.

6. Means for quantification

We introduced two quality measures to evaluate obtained semantic vectors: (1) clustering consistency and (2) system-to-human correlation.

6.1. Clustering consistency

This measure is based on the organization of the 80 object classes contained in the COCO dataset into eleven super-categories, like animal, vehicle, kitchenware, etc [24]. In an ideal case, the closest neighbors of a given semantic vector would all belong to the same super-category. We measure to which proportion this is satisfied. Specifically, for every object o in a super-category, we consider its semantic vector and then find the k nearest neighbor semantic vectors. Of those k vectors we count the number of objects that fall into the same super-category as o and normalize the count by k. Then we average results for all o in that super-category, for all super categories and, finally, we average scores for $k = 1$ to $k = 5$ (5 is the size of the smallest super-category in COCO) to obtain a single scalar score. For COCO we have 80 object classes in an 80-dim semantic vector space. To remain comparable, we are also evaluating LabelMe using a compatible subset of 60 classes occurring in LabelMe from the 80 COCO classes, but now the semantic vectors have 3288 instead of 80 dimensions. The clustering consistency measure is based on similar grounds as the most common standard classification accuracy metrics for a k-nn classifier, like e.g. precision and recall. To keep the main text concise, we discuss this relation in more detail in subsection A.4 in the Appendix.

6.2. System-to-human correlation

For this we compared similarity (relatedness) scores of word-pairs determined by human raters, using the benchmarks MEN, Simlex, and Scene250, with ratings for the same pairs calculated by the different automatic procedures. This can be best understood by a simple example.

Let us assume that our benchmark dataset consists of five word-pairs, which are here listed in descending order according to their human-determined similarity (rank-order list):

motorcycle-bicycle, train-motorcycle, car-train, train-orange, orange-airplane.

From the vector representations of our concepts, calculated by – say – *VGG16*, we can compute the semantic vector *distance* between word pairs. Semantic distances could be: car-train: $d = 0.2$, orange-airplane: $d = 0.8$, motorcycle-bicycle: $d = 0.1$, train-orange: $d = 0.9$ and train-motorcycle: $d = 0.3$. For comparison, we – thus – use *inverse distance* and we get the following rank-ordered list:

motorcycle-bicycle, car-train, train-motorcycle, orange-airplane, train-orange.

Both lists are not identical but clearly correlated, which we can quantify using the Spearman rank correlation coefficient, for which we get in this example a value of 0.8.

This Spearman rank correlation calculation has been performed for *COn*, *CCn*, *COs*, *CCs*, *InceptionV1*, *InceptionV4*, *VGG16*, *ResNet50*, *Skip-Gram* and *Random* vectors against MEN, Simlex, and Scene250, whenever there were enough word-pairs existing in *both* datasets (see section A.3 for quantification of word-pair intersections).

Note that all automatic procedures will calculate only one distance value for any given pair. Thus, we compared the resulting inverse distance rank list to *both*, similarity as well as relatedness, rank lists from the human raters.

7. Experiments

This section reports the quantitative results of our experiments. Regarding the measure of semantic distance, we found that there are no consistent differences between results obtained with Euclidean versus cosine distance. Both measures

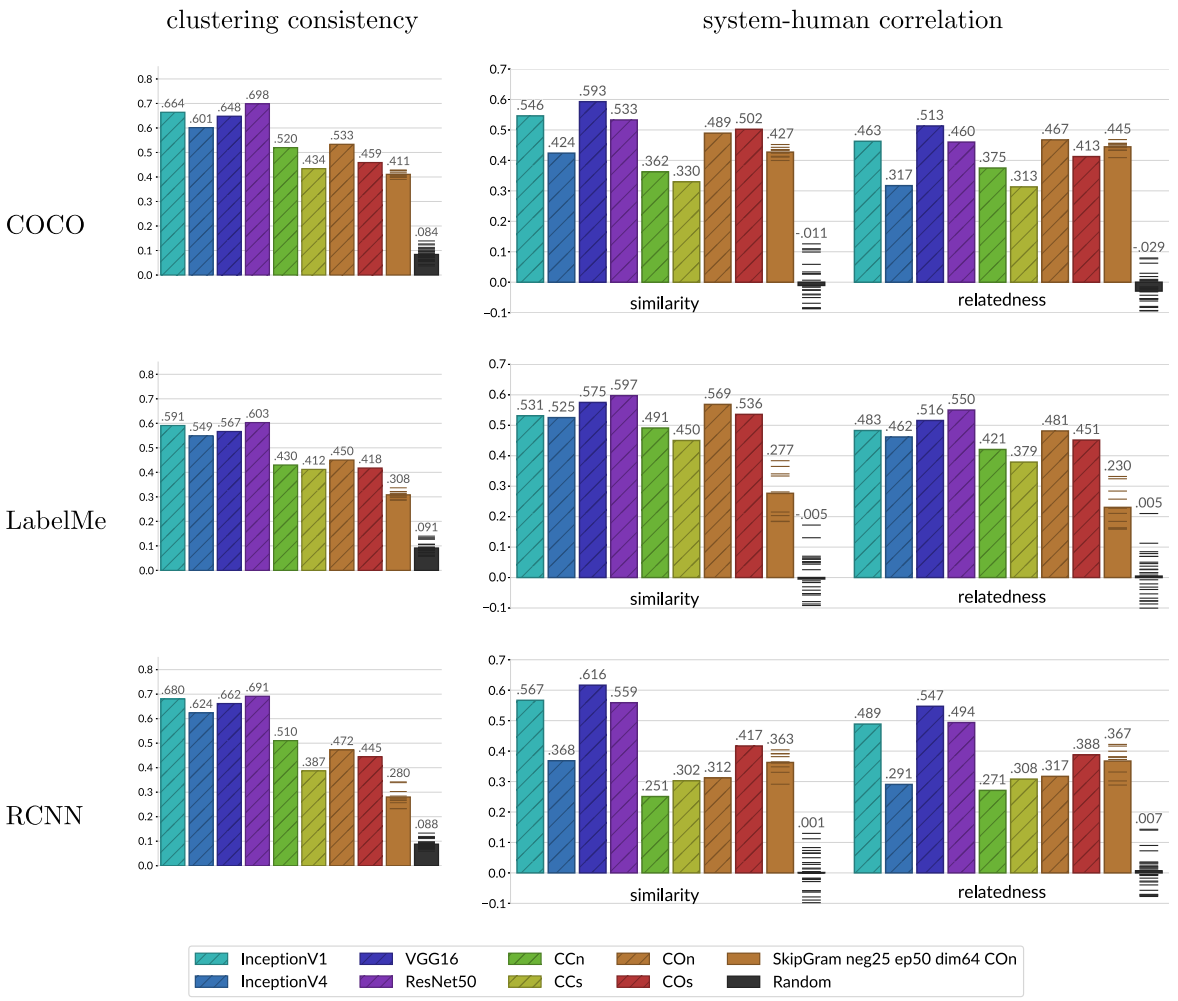

Fig. 4. Experimental evaluation of different context models (given by different colors, see legend at the bottom of the figure) on COCO (top), LabelMe (middle) and RCNN (bottom). On the left side clustering consistency is shown. On the right system-to-human correlation for Scene250 data measuring similarity (sim) and relatedness (rel) are shown.

render the same results with usually less than ±10% differences. Therefore, we show only results for the cosine distance, which is the one commonly used in the relevant literature.

In the following we show that our visual-feature based semantic vectors match the quality of state-of-the-art text-based methods for everyday objects despite being extracted from different data with a much smaller number of samples.

7.1. Comparison of context models

The first experiment (see Fig. 4) is used to compare different context models and here first and foremost label-based context vs. visual feature-based context. Performance is analyzed using semantic vectors that we had we obtained from the three scene datasets (from top to bottom): COCO, LabelMe and RCNN. Number of classes, scenes and contexts in those datasets are provided in Table 2 in section 5.1. Label-based contexts are annotated as *CO* or *CC* (corresponding to Co-Occurrence and Co-Count as discussed in section 3.1). With subscripts *s* or *n* we indicate whether the reference objects is excluded from the context description (s means self-included, n means non-self-included). Visual-feature-based contexts are obtained by the CNNs *VGG16*, *InceptionV1*, *InceptionV4* and *ResNet50*. All the mentioned contexts are obtained using averaging for context integration. Skip-gram performance for the context COn is provided as a baseline, for comparison with the averaging-based methods. Chance performance is indicated for each case, too (see label "Random"). For the skip-gram and the chance columns in Fig. 4 the ten samples, from which the average is obtained, are shown by horizontal lines on the corresponding column. The remaining columns do not rely on any randomized sampling.

All plots on the left show the *clustering consistency* measure for the different methods and the ones in the center and to the right show the *system-to-human correlation* as described above. For *clustering consistency*, neighbors in the semantic vector space are analyzed based on the existing division of the 80 COCO object classes into 11 super-categories as given

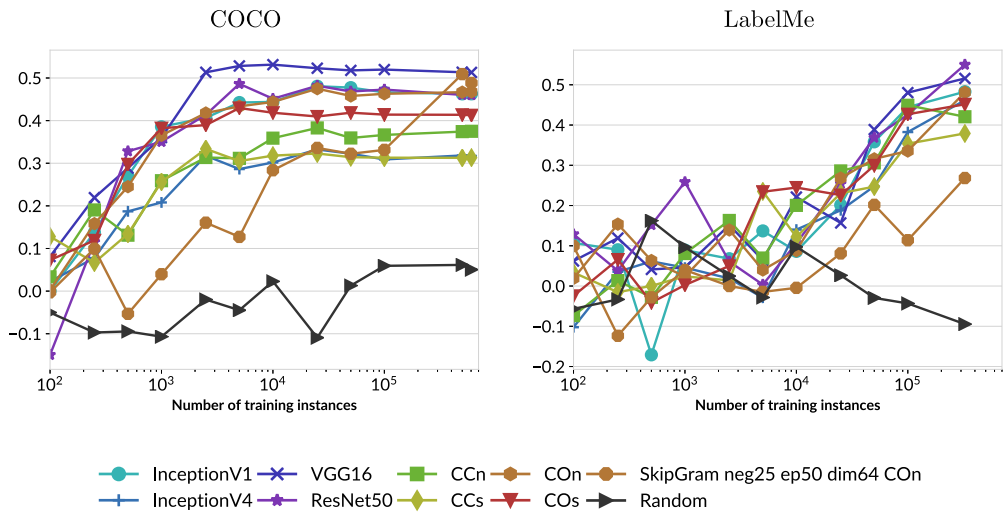

Fig. 5. Plot of the Spearman correlation coefficient with similarity ratings in relation to the number of instances used for training.

by the developers of this dataset. System-to-human correlation is based on our newly introduced Scene250 benchmark. One can observe that all measures for all contexts show clear above-chance performance. Different measures show similar rank-order for the analyzed contexts, which also holds between the COCO and LabelMe datasets. RCNN shows a bit lower scores when measuring system-to-human correlation for label-based contexts. However, RCNN is the one dataset with the sparsest labels: only 2.5 labels per image on average as reported in the method section above. Apparently, object density is still enough to support visual-feature-based context analysis, but is too low for reliably analyzing label co-occurrences.

Furthermore, we find that the skip-gram negative sampling baseline (based on COn, see 9-th column in each plot) is performing almost always worse than the corresponding COn context obtained through simple averaging (7th column). The exception is the RCNN dataset where sparsity of labels seem to make evaluation of label-based contexts less stable, as discussed in the paragraph above. Relatively low skip-gram performance might have several reasons: In contrast to text corpora where skip-grams are employed on very big datasets, our dataset have a fairly small amount of labels. Also, negative sampling requires many hyper-parameter choices (number of negative samples, sampling strategy of negatives) and there is some chance that there are better configurations which we did not find despite a fairly extensive search (see subsection A.6 in the Appendix).

Surprisingly we found that visual feature-based contexts (specifically, look at *ResNet50*) exhibit better performance than label-based ones. Hence, it appears that CNN-generated visual features have a stronger association with the meaning of objects than manually assigned class labels, where – in addition – the latter is also more expensive to obtain. In general, best performance is found for context model *CO*(*s* and *n*) and for feature-based model *ResNet50* and *VGG16*. Thus, for further evaluations we put an emphasis on these methods.

Concerning *system-to-human correlation* (central and right column) we find that our system most of the times produces slightly bigger system-to-human correlations for similarity than relatedness. We get a maximal correlation of 0.616 for similarity and of 0.550 for relatedness. Correlation *between* humans is 0.544 for similarity and 0.659 for relatedness. This is to some degree remarkable because it shows that this system performs very close to the human level.

An additional, important observation is that the RCCN dataset, which is automatically labeled, together with neural net methods (visual-feature-based approaches (*VGG16*, *InceptionV1*, *ResNet50*, left bars) will lead to scores that are even higher than those from the COCO dataset. This is quite remarkable as automatic labeling using Mask R-CNN is less reliable than human-labeling. These findings demonstrate the impressive progress in object detection in recent years and suggest that applying an object detector as a pre-processing step is a viable solution towards a system that handles a huge number of scenes, beyond a quantity that can annotated by humans. In the future, performance could be improved simply by plugging-in a more advanced object detection algorithm. In particular it would be interesting to see a larger set of objects being covered.

7.2. Scale: dependency on the size of the dataset

Under the assumption that quality increases with number of data points, a valid question is to ask whether quality has already reached ceiling with the here-used number of scenes or whether we can expect it to further increase given a larger dataset. Since the number of considered object classes might be a crucial factor, we determine the scaling behavior for the COCO dataset as well as for the more diverse LabelMe dataset.

Fig. 5 suggests a different answer for both datasets: the results that have been obtained from the COCO dataset (80 different objects and 604,907 context instances (see subsection 3.1)) reach ceiling already at 1000 analyzed contexts (not

Table 3
Fusion of contexts extracted from COCO and from RCNN.

	Clustering consistency	Scene250	
		sim	rel
MSCOCO (80 concepts)			
COn	0.533	0.467	0.489
VGG16	0.648	0.513	0.593
ResNet50	0.698	0.460	0.533
Late Norm: VGG16+COn	0.680	0.643	0.685
Early Norm: VGG16+COn	0.656	0.579	0.645
Late Norm: ResNet50+COn	**0.738**	**0.663**	**0.696**
Early Norm: ResNet50+COn	0.702	0.522	0.587
RCNN (79 concepts)			
COn	0.472	0.317	0.312
VGG16	0.662	0.547	0.616
ResNet50	0.691	0.494	0.559
Late Norm: VGG16+COn	0.717	0.653	0.704
Early Norm: VGG16+COn	0.690	0.614	0.681
Late Norm: ResNet50+COn	**0.726**	**0.686**	**0.715**
Early Norm: ResNet50+COn	0.713	0.554	0.614

scenes) while, when using LabelMe (3288 different objects), there is no clear saturation visible even at 325,288 analyzed instances, which are all the instances available for this dataset. This can be expected since the number of context vector dimensions for LabelMe is much larger (due to a higher number of possible contextual objects) and this allows for more configurations of the context, requiring more examples for convergence.

7.3. Fusion of contexts

Table 3 reports fusion results on the supervised COCO dataset as well as the automatically annotated RCNN dataset. We apply fusion of the context COn (which is among the best label based contexts as shown in Fig. 4) with visual-feature based contexts VGG16 and ResNet50. It can be seen that fusing of the two types of contexts improves both clustering consistency as well as system-to-human correlation both for similarity and relatedness, in both datasets. Hence, fusion turns out to be an inexpensive yet powerful method to increase quality. Late fusion provides consistently better scores than early fusion. Thus, we will be using late fusion further, in comparing our methods to the state-of-the-art.

7.4. Comparison to state-of-the-art

How does our image-based method compare to automatic text-based methods from NLP such as Word2Vec, Glove, etc.? This comparison is made in two ways: 1) We determine and compare clustering consistency for the different methods and 2) we ask how do all the methods behave relative to the human-rater based ground-truth data from the MEN, Simlex999, and Scene250 datasets. Hence, for this we ask specifically, whether the Spearman correlation coefficient between image-based vs. human rater based is bigger or smaller than the one between NLP-based vs. human rater-based.

For comparison we use the purely text-based approaches Word2Vec skip-gram negative sampling trained on Google news [28] (W2V-Neg) and Glove 6B [29] trained on Wikipedia and Gigaword 5 (*Glove*). We also use multi-modal text- and image-based methods proposed by Kiela and Bottou [21] (*KielaMax* and *KielaMean*).

This is compared to our methods where we use as a basis *ResNet50* and *VGG16*, which are the best performing methods according to previous experiments and extend the comparisons to different combinations of these base-methods by using Late Fusion with co-occurrence context without self-counting (*COn*) as well as in addition Late Fusion with the NLP-method skip-gram negative sampling whose vectors were computed on Google News (*W2V-Neg*).

Note that any considered comparison could only be made for the smallest common subset of concepts (or word pairs), which corresponds to the respective intersection of the datasets used in that comparison. This is always an intersection between three sets, for example COCO with Scene250 and the data used in Glove, etc. In section A.3 in the Appendix we explain, how the intersection subsets look like. Due to the fact that RCCN is a descendent of COCO, their intersections are always the same.

Table 4 shows the results.

7.4.1. Clustering consistency

We evaluate clustering consistency on the intersection of categories between scene datasets (COCO, LabelMe, or RCNN) and state-of-the-art implementations (*Glove, W2V-Neg* and *Kiela*). As before, we use the eleven super-categories defined in

Table 4
Comparison of our representation with state-of-the-art methods on various benchmarks, from left to right: Clustering consistency, Spearman correlation for relatedness (rel) and similarity (sim) ratings using the Scene250 benchmark, Spearman correlation for similarity using SimLex999 and for relatedness using MEN.

	Clustering Consistency	Scene250		SimLex999	MEN
		sim	rel		
COCO	64 concepts	165 pairs		9 pairs	3 pairs
ResNet50	0.682	0.459	0.548	–	–
LateFusionNorm: ResNet50+COn	0.670	0.702	0.744	–	–
LateFusionNorm: ResNet50+COn+W2V-Neg	0.693	**0.726**	**0.766**	–	–
VGG16	0.631	0.537	0.626	–	–
LateFusionNorm: VGG16+COn	0.654	0.689	0.740	–	–
LateFusionNorm: VGG16+COn+W2V-Neg	0.678	0.703	0.754	–	–
Glove	0.590	0.666	0.744	–	–
W2V-Neg	**0.716**	0.707	0.742	–	–
KielaMax	0.580	0.435	0.494	–	–
KielaMean	0.592	0.520	0.574	–	–
LabelMe	60 concepts	151 pairs		166 pairs	626 pairs
ResNet50	0.641	0.588	0.617	0.271	0.343
LateFusionNorm: ResNet50+COn	0.647	0.681	0.727	0.230	0.431
LateFusionNorm: ResNet50+COn+W2V-Neg	0.663	0.711	**0.755**	0.290	0.545
VGG16	0.606	0.567	0.606	0.297	0.354
LateFusionNorm: VGG16+COn	0.614	0.666	0.716	0.241	0.404
LateFusionNorm: VGG16+COn+W2V-Neg	0.622	0.687	0.737	0.268	0.461
Glove	0.587	0.704	0.752	0.289	0.679
W2V-Neg	**0.715**	**0.727**	0.751	**0.463**	**0.776**
KielaMax	0.589	0.462	0.506	0.430	0.573
KielaMean	0.599	0.549	0.598	0.366	0.608
RCNN	64 concepts	165 pairs		9 pairs	3 pairs
ResNet50	0.631	0.521	0.590	–	–
LateFusionNorm: ResNet50+COn	0.677	0.724	0.754	–	–
LateFusionNorm: ResNet50+COn+W2V-Neg	**0.719**	**0.743**	**0.772**	–	–
VGG16	0.605	0.591	0.661	–	–
LateFusionNorm: VGG16+COn	0.677	0.708	0.751	–	–
LateFusionNorm: VGG16+COn+W2V-Neg	0.703	0.718	0.763	–	–
Glove	0.590	0.666	0.744	–	–
W2V-Neg	0.716	0.707	0.742	–	–
KielaMax	0.580	0.435	0.494	–	–
KielaMean	0.592	0.520	0.574	–	–

COCO. Under these assumptions, we find that there is an intersection of 60 and 64 concepts, respectively, between the datasets for these cases.

Fusing label-based with feature-based contexts (*VGG16+COn*) improves results. Adding a text-based context (*VGG16+COn+W2V-Neg*) further increases performance, although this improvement is sometimes rather small. This confirms our findings about the usefulness of context fusion from above, now also including fusion with text-based contexts.

Concerning pure NLP-based methods we find that *W2V-Neg* shows a very strong performance, which our methods alone do not achieve. But we are not far off and still beat any of the other NLP-based methods.

7.4.2. System-to-human correlation

System-to-human-correlation is measured relative to the Scene250, SimLex999 and MEN datasets. The table shows that – when considering COCO (or RCNN) – there is essentially no intersection between the data (9 and 3 pairs, respectively). Scene250 had been purposefully created so that there was a useful overlap (165 and 151 pairs).

Findings on the Scene250 benchmark are closely following the results obtained using clustering consistency. However here the fusion of visual contexts with the *W2V-neg* was twice outperforming pure *W2V-neg*, which happened for the scene datasets with strictly controlled object labels (COCO and RCNN).

The overlap between LabelMe and SimLex999 (166 pairs) as well as MEN (625 pairs) was big enough. This intersection, however, contains a big proportion of very general objects, which do not have straightforward visual representation (examples of those are *art, bedroom, construction, game, image, reflection, shoulder, smile, subway, sunlight, water*). Also, it includes super-categories, e.g., *animal, cloth, container, food, furniture, vehicle* the usage of which introduces ambiguities in image an-

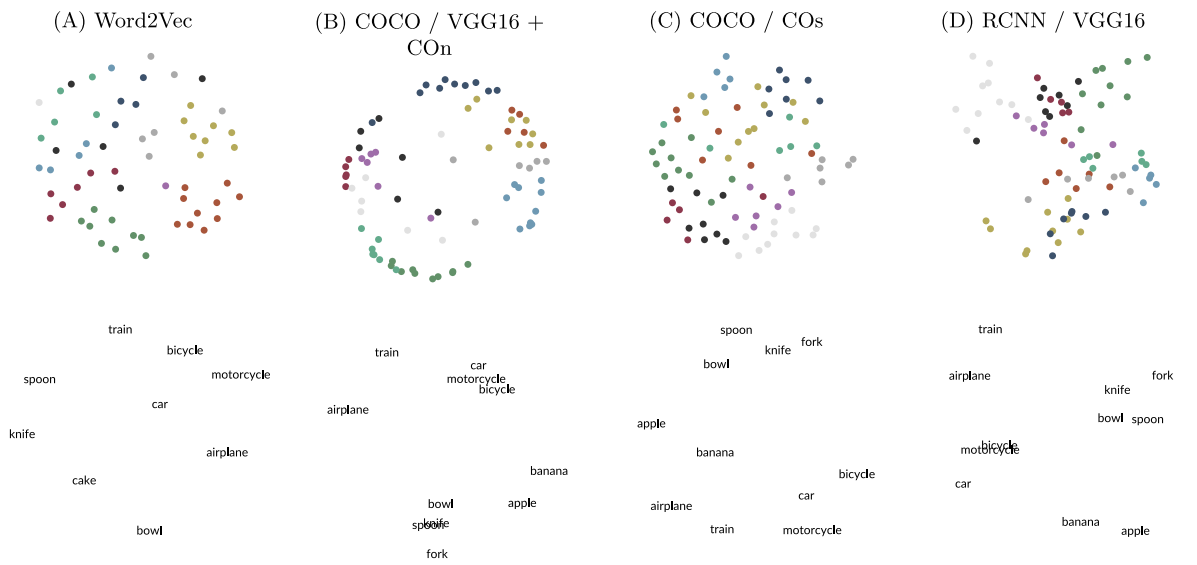

Fig. 6. Qualitative results, top: Plot of 80 object vectors of labels in COCO projected to 2D using multi-dimensional scaling with color corresponding to super-category. Bottom: Plot of some belonging word labels.

notations (e.g., is it a bus or a vehicle?). Some pairs in the benchmarks include words with multiple meanings (homonyms) where specifically the non-object meaning makes the words related, like (guitar, rock) or (card, bridge). As a consequence, here our image-based methods are worse than the NLP-based ones.

Another reason for this is based on the statistics. LabelMe has only 28,072 scenes were – for robustness reasons – we had considered only objects that were labeled in the scenes three times or more. Thus, some objects we included into our analysis were fairly infrequent (or appeared in contexts infrequently). This likely led to too thin data to gain statistically solid scene-based knowledge about those more abstract objects mentioned above. Furthermore, SimLex999 and MEN were collected to assess general concepts are therefore not the best to judge object concepts.

Scene250, instead, had been created specifically to judge everyday object pairs. Pairs were rated by informed subjects who treated word-pairs very thoughtfully. It seems that under this more rigorous condition differences between image-based and word pair-based assessments vanish.

Summary. Summed up, our methods compare very well to text-based semantic vectors and often even outperform multi-modal semantic algorithms as long as the evaluation is run over a set of proper objects. W2V-Neg turns out to exhibit very good scores, but note that the text corpora that W2V-Neg was trained on are magnitudes larger both in terms of vocabulary size (which would be similar to our context) and number of words (object instances) than the datasets we employ.

7.5. Qualitative results

Results can be visualized by reducing their dimensions to two while preserving distances using multi-dimensional scaling. This way, qualitative insights can be obtained. Here we present three different types of visualization:

- **Super classes (Fig. 6 Top)**: A plot of all vectors corresponding to COCO classes visualized as dots with super-categories (e.g. animal) determining their color. Ideally, object vectors pertaining to the same super-category should form groups.
- **Local space (Fig. 6 Bottom)**: On a more fine-grained scale, object vectors from a pre-selected set of names are plotted. Instead of a dot, here the respective object name is plotted.
- **Functional (Fig. 7)**: Instead of grouping based on super-categories, we split the set of concepts according to a functional trait. Concepts that have this trait are visualized in color together with their name while concepts that do not have a trait are visualized as gray dots. We defined the traits `larger_than_human`, `sharp`, `round`, `has_wheels` and `can_fly`.

We observe that concepts indeed cluster according to their super-category and see reasonable groups of objects in the local space. Furthermore, we find concepts being grouped together within functional groups, even though they do not share the same super-category. Thus, these figures provide a visual confirmation of the above reported clustering results.

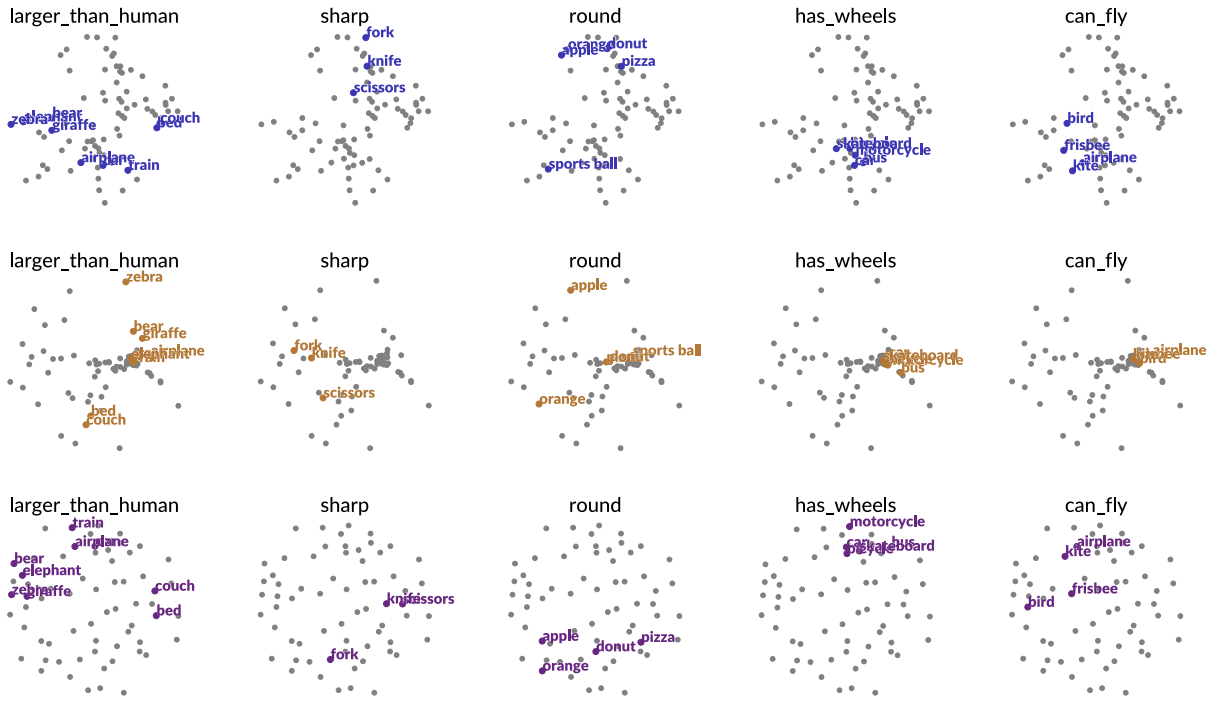

Fig. 7. Qualitative analysis of how well functional concepts are represented. Top: VGG16, middle: COn, bottom: W2V-Neg. Positive examples for the categories are highlighted in red while other concepts are gray points.

8. Discussion

8.1. Main contribution

In this study we were asking whether the distributional hypothesis of linguistics [17] would also hold for images. Text is a linear structure, while images are 2D-projections of 3D scenes and this higher dimensionality might lead to a dilution of information. Thus, we specifically wanted to quantify how powerful a semantic vector representation obtained from scene information still is. Here, we presented a simple but effective method to approach the problem of understanding objects in their scene context by extracting semantic vectors. The main result of this study is that across many experiments, methods and data, text and images appear equally powerful as sources for context analysis. This is to some degree astonishing, because text-based methods, like W2V-Neg and GloVe, use more than 10^{11} words, while we have used on the order of 10^5 images with 10^6 labels only.

Taken together this shows that visual context contributes to our cognitive concepts of objects and should be considered when building semantic vectors. Our evaluations also suggests that simple integration methods like the one presented in this paper suffice to capture this kind of semantics. The amount of data is more critical than the (text- or image-based) context method used.

8.2. Pros and Cons

Why should one analyze images instead of (or together with) text? One aspect seems to be the above mentioned density of information. Different from our own expectations, it seems that even a small set of images can already carry substantial amounts of (certain) context information. This holds specifically for context information on aspects, which texts don't talk about, like the layout of the utensils in a kitchen drawer or the way a workshop might be arranged. Arguably, not many texts provide this type of information and in these cases context extraction should, thus, be easier in images than in text. Action understanding and execution in robotics has encountered this situation, too. While it is easy to obtain massive higher level action information from text sources, some types of information, which can be vital for a household robot, are missing. For example: Hardly ever one finds a text on how to handle a knife or a spatula, when cooking. Videos and images, on the other hand, are abundant.

In addition, to this another important beneficial aspect when using image-based context extraction is that our results on the performance of RCNN-labeled as well as CNN activation-determined object context indicate that it is also possible to perform automatic semantic assessment of objects in scenes without human labeling, where accuracy is similar to that of the human-supervised methods.

Furthermore, we found in the small pilot experiment shown in Fig. 7 that even functional groupings can be extracted. Other visual feature-based grouping might be possibly, too, but a more detailed analysis is currently not possible mainly because object categories are too broad (from the category "knife" we cannot tell if it is metallic or not). For a quantitative analysis it would be interesting to train a classifier or regressor to capture specific aspects of the semantic vectors. However, this is currently not possible since the set of objects is still too small and the number of features too large. E.g. training something like "eatable" on maybe tens of positive samples and hundreds of negative samples while having thousands of features will likely overfit even with simple classifiers. At the moment, this prevents rigorous quantitative analyses of this kind, but the results from Fig. 7 look promising and future work (after addressing the above mentioned problems) should be possible in this direction.

Image-based context analysis, however, has clear disadvantages, too. For example, abstract concepts cannot be extracted. Also, longer text-narratives often contain aspects of reasoning. Images do not. Hence, artificial narrative generation from images, at least so far, remains relatively shallow as compared to the creativity with which – for example – a child describes an image. This is achieved by humans being able to extract in a generative way complex context from an image. Modern image analysis methods, including the ones presented here, stop still short of this.

However, from a practical perspective we are confident that existing methods to extract word vectors, such as Glove and Word2Vec, that serve as a basis for numerous methods in NLP, could benefit from the addition of visual, scenic context. This would require either a large dataset or an object detector capable of handling thousands of classes, but considering the recent progress in computer vision it is only a matter of time until these become available.

8.3. Other aspects and future work

It may be interesting to discuss these findings also from a different perspective. "Meaning" cannot be obtained by considering nothing but text. This issue is commonly known as the symbol grounding problem [16] in artificial intelligence: Glenberg and Robertson [15] argue that it is not possible to assign meaning to an abstract symbol like a word as long as it is expressed only with respect to other abstract symbols. This is the case for text but also for labels of image annotations. However, these authors suggest overcoming this problem by *linking* different modalities, where here we had considered text and images.

Evidently humans often perform co-context analyses of text and image information, too; for example, where text remains too abstract and opaque without an image. A good example of this are industrial instruction sheets for small product assembly by human workers. Text-only instructions are here often quite nontransparent. One example we had worked on in the context of the European funded ACAT (Action Categories) project had been: "Place the rotor cap on top of the magnet holder". This instruction becomes only meaningful when seeing the actual objects (before and after this assembly step).

Object class vectors, in conjunction with text, might afford this to some degree as they have been obtained from two complementary modalities. This way, an object's spatial context becomes linked with its textual context. Hence, the process of acquiring meaning would now stand on two legs. While we cannot directly show this, our results support at least the notion that context is strengthened by such combinations.

Several additional aspects could be addressed in future work. Currently, the knowledge captured by the model is expressed by linear translations in a high dimensional feature space. It might be useful to investigate more complex representations, e.g. to reduce the size of the feature space. As a starting point hyperbolic geometry could be considered. Additionally, we have indeed performed some early experiments concerning the combination of vectors spaces obtained from different modalities. The performance gain in these experiments indicates that this direction might be promising as it is a fairly cheap way for improving results relative to single modal semantic vectors.

All this could, for example, lead to an improved identification of task-relevant objects to suggest tasks and to generate planning problem definitions automatically depending on the objects in the scene. This is of great use in robotic executions of human-like tasks [1], where, for instance, the automatic generation of task descriptions can significantly ease the human-robot interaction.

Acknowledgements

This research has received funding from the European Commission Horizon 2020 Program H2020-ICT-2016-1 under grant agreement 731761 "IMAGINE".

Appendix A

A.1. Derivation of co-count context

This derivation explains the co-count context. Co-occurrence context can be seen as a special case of this with the frequency of any object class being limited to one. Hence, the argument following now can be modified accordingly and transferred to these other cases, too.

A.1.1. Probabilistic derivation

A common way to describe events without knowing the number of trials is the Poisson distribution [3]. The choice of the Poisson distribution is justified because the occurrence of objects can be considered as events in the spatial domain. Furthermore, a Poisson distributed random variable takes only positive integer values, which is true for co-occurring objects. Hence we will assume that the frequency of an object m occurring in the context of object k follows a Poisson probability distribution with rate λ_k^m. This means, to generate a context vector for an object, we would draw samples from a Poisson distribution with specific parameters for each component. Following the idea of the distributional hypothesis, as these parameters describe the context, they will be used to represent the object classes. Subsequently, we show how to obtain these parameters.

Having observed N_k instances of each object class k, the variable $i \in [1, N_k]$ refers to the context vector $\hat{\mathbf{c}}_{ki}$ of the i-th instance of class k. We assume a probabilistic model $P(\mathbf{c} \mid o)$ in which the object class determines the context. The goal is to find parameters $\mathbf{\Lambda} = \{\lambda_k^m : k, m \in [1, K]\}$ that maximize the likelihood L of these observations.

$$L(\mathbf{\Lambda}) = P(\mathbf{c} \mid o; \mathbf{\Lambda}) = \prod_{k=1}^{K} \prod_{i=1}^{N_k} P(\mathbf{c} = \hat{\mathbf{c}}_{ki} \mid o = k; \mathbf{\Lambda}). \tag{2}$$

Due to monotonicity, a logarithm can be applied. This facilitates further steps and retains the same maximum.

$$\log L(\mathbf{\Lambda}) = \sum_{k=1}^{K} \sum_{i=1}^{N_k} \log P(\mathbf{c} = \hat{\mathbf{c}}_{ki} \mid o = k; \mathbf{\Lambda}). \tag{3}$$

The Poisson distribution's mass function is inserted. Due to the assumption of co-occurring objects being independent, the occurrence probabilities of context objects follow a multiplication:

$$P(\mathbf{c} = \hat{\mathbf{c}}_{ki} \mid o = k; \mathbf{\Lambda}) = \prod_{m=1}^{K} \frac{(\lambda_k^m)^{\hat{c}_{ki}^m}}{\hat{c}_{ki}^m!} e^{-\lambda_k^m}. \tag{4}$$

In order to find the maximum likelihood, the derivative with respect to $\boldsymbol{\lambda}_k$ is calculated,

$$\log L(\mathbf{\Lambda}) = \sum_{k=1}^{K} \sum_{i=1}^{N_k} \sum_{m=1}^{K} \hat{c}_{ki}^m \log \lambda_k^m - \lambda_k^m - \log \hat{c}_{ki}^m!, \tag{5}$$

$$\frac{\partial \log L(\mathbf{\Lambda})}{\partial \lambda_k^m} = \sum_{i=1}^{N_k} \frac{\hat{c}_{ki}^m}{\lambda_k^m} - 1. \tag{6}$$

Since $\mathbf{c}$ refers to positive co-occurring object counts and λ_k^m describes a rate that is positive, the second derivative must be negative indicating a maximum. Hence, by setting the first derivative zero, we obtain the optimal parameterization $\boldsymbol{\lambda}_k^*$. Simplified, $\boldsymbol{\lambda}_k^*$ is the average of all context vectors:

$$\boldsymbol{\lambda}_k^* = \left(\arg\max_{\mathbf{\Lambda}} L(\mathbf{\Lambda}) \right)_k = \frac{1}{N_k} \sum_{i=1}^{N_k} \hat{\mathbf{c}}_{ki}. \tag{7}$$

A.1.2. Geometric perspective

Rather than considering observations of objects as events, the problem also permits a geometric perspective, with observations of context vectors involving co-occurrence frequencies being spread in a high-dimensional (K dimensions) vector space. Every observed context has an associated class and the goal is to find a representative context for each class. We can interpret this problem as optimization of a cost function. If the cost is defined as the sum of the mean quadratic Euclidean distances of each class, the minimum again can be analytically derived to be the mean context:

$$\boldsymbol{\lambda}_k^* = \arg\min_{\mathbf{c}_k} \frac{1}{N_k} \sum_{i=1}^{N_k} (\mathbf{c}_k - \hat{\mathbf{c}}_i)^2 = \frac{1}{N_k} \sum_{i=1}^{N_k} \hat{\mathbf{c}}_i \tag{8}$$

A.2. Alternative context integration methods

In the main text we had compared average context with Skip Gram context integration. Many more integration methods would be possible, too. For example, using the median appears a possible natural choice. Thus, here we take a look at this based on the COCO dataset (Fig. 8) and compare its performance with averaging (see Fig. 8). In each plot the first four columns are obtained by averaging, while the columns 5 to 8 are obtained with the median. The results show that

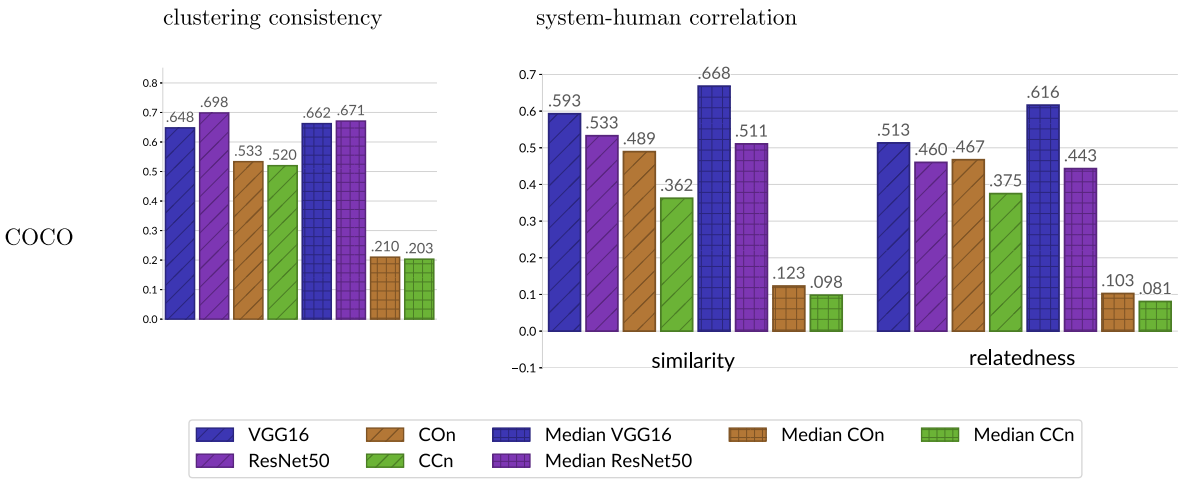

Fig. 8. Comparison between averaging (diagonal hatching) and median (square hatching) for context integration.

the median sometimes leads to an improvement and sometimes it decreases performance. In particular in case of VGG16 the median yielded a fairly strong improvement. However, as the median does not show stable performance (decrease in performance may be substantial too, e.g. for COn and CCn context methods in the figure), averaging remains a better method.

A.3. Comparison of fusion methods

There are multiple ways for combining different types of context vectors for a concept. In the main text we analyzed one type of early fusion (concatenating context instances) and one type of late fusion (concatenating semantic vectors). We applied one specific type of normalization for early fusion and one type for late fusion. Here we will analyze more such methods.

The investigated methods are concatenation-based (except for sum) and vary in the way how normalization is carried out prior to concatenation. Let us say $\mathbf{a}$ and $\mathbf{b}$ stand for two vectors and $||$ the concatenation operation. We define the following fusion methods of those vectors:

1) Simple concatenation:

$$cat(\mathbf{a}, \mathbf{b}) = \mathbf{a} \,||\, \mathbf{b}. \tag{9}$$

2) Simple averaging:

$$\text{average}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} + \mathbf{b}}{2}.$$

Note, however, that this method is fairly limited as it requires the context vectors to have the same length.

3) Concatenation with length-based normalization:

$$\text{eq}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a}}{dim(\mathbf{a})} \,||\, \frac{\mathbf{b}}{dim(\mathbf{b})},$$

where dim denotes the length of the vectors.

4) Early fusion with weight normalization (same as in the main text):

$$\text{norm}_{early}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} - \mu_{\mathbf{a}}}{\sigma_{\mathbf{a}}} \,||\, \frac{\mathbf{b} - \mu_{\mathbf{b}}}{\sigma_{\mathbf{b}}},$$

where μ_a, μ_b and σ_a, σ_b are scalar mean and standard deviation of vectors a and b respectively.

5) Late fusion with weight normalization (same as in the main text):

$$\text{norm}_{late}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} - \mu(A)}{\sigma(A)} || \frac{\mathbf{b} - \mu(B)}{\sigma(B)},$$

considers vectors in their respective embedding (A and B) and computes mean and standard deviation independently for every dimension, i.e. here μ and σ are vectors

Table 5
Comparison of various fusion methods on COCO.

	Clustering	Scene250	
		sim	rel
CCn	0.520	0.375	0.362
COn	0.533	0.467	0.489
(1) Early Concat: CCn+COn	0.529	0.385	0.377
(2) Early Average: CCn+COn	0.530	0.395	0.391
(1) Early Concat: InceptionV1+COn	0.670	0.466	0.550
(3) Early Equal: InceptionV1+COn	0.627	0.573	0.611
(4) Early Norm: InceptionV1+COn	0.682	0.573	0.637
(5) Late Norm: InceptionV1+COn	**0.693**	**0.643**	**0.674**
(3) Late Equal: InceptionV1+COn	0.627	0.573	0.611
(7) Late NormToOne: InceptionV1+COn	0.535	0.467	0.490
(6) Late NormStdOnly: InceptionV1+COn	0.677	0.419	0.499
(1) Early Concat: VGG16+COn	0.649	0.513	0.593
(3) Early Equal: VGG16+COn	0.639	0.563	0.615
(4) Early Norm: VGG16+COn	0.656	0.579	0.645
(5) Late Norm: VGG16+COn	**0.680**	**0.643**	**0.685**
(3) Late Equal: VGG16+COn	0.639	0.563	0.615
(7) Late NormToOne: VGG16+COn	0.535	0.468	0.490
(6) Late NormStdOnly: VGG16+COn	0.645	0.473	0.556

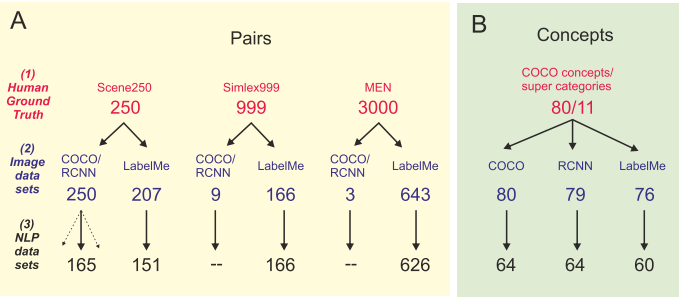

Fig. 9. Intersections between the different data sets.

6) A simplified version of weight normalization only considers the standard deviation:

$$\text{norm}_{std}(\mathbf{a},\mathbf{b}) = \frac{\mathbf{a}}{\sigma(A)} || \frac{\mathbf{b}}{\sigma(B)}.$$

7) Late fusion norm to one concatenates vectors that were normalized to length one.

$$\text{norm}_{one}(\mathbf{a},\mathbf{b}) = \frac{\mathbf{a}}{|\mathbf{a}|} || \frac{\mathbf{b}}{|\mathbf{b}|}.$$

Note, when using context averaging as integration method, early fusion and late fusion are equivalent for concatenation, average and length normalization, because the order of the operations are commutative.

In Table 5 results obtained using different fusion methods are shown. First we compare results for simple concatenation and averaging between contexts CCn and COn. Note, we cannot use averaging for other, potentially more interesting context combinations, as for that vectors shall be of equal length. Here we see that none of the two indicated context fusion methods is of substantial advantage.

We experimented also with quite a few other fusion methods for different types of context vectors: label-based and visual-feature-based. In summary this shows that fusion will usually improve the result, where here the "Late Norm" method is the best. Note however that also for fusion other methods would be possible and an exhaustive analysis cannot be performed.

A.4. Intersections of different datasets

When evaluating our method against the state-of-the-art methods on given ground-truth benchmarks, we needed to consider a tripartite intersection of different datasets (1, 2, 3, numbering as in Fig. 9, left side). Here we will explain the reasoning behind this and specify coverage for these intersections.

Concerning similarity ratings (Fig. 9 A), we compared (3) conventional NLP text-based methods with (2) our image-based methods, grounding these comparisons on (1) three human-rated datasets: Scene250, Simplex999 and MEN. These datasets

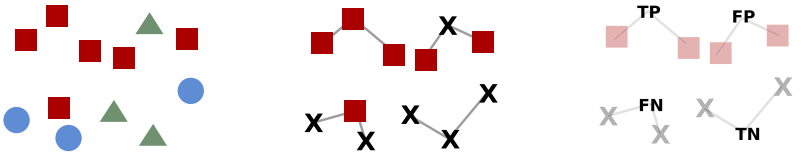

Fig. 10. Sketch of the precision/recall evaluation: Left concepts with super-category, middle: square centered view with two-nearest-neighbors of four concepts being indicated, right: error categories based on the true labels and the classification based on two nearest neighbors.

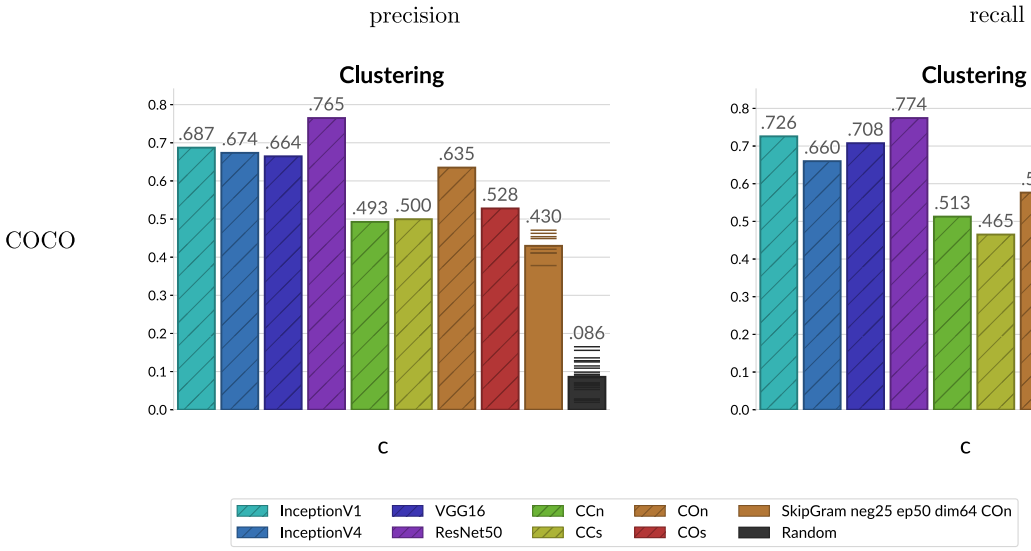

Fig. 11. kNN scores for various context extraction methods.

are named in the top row in Fig. 9 (red). The human rated datasets are a collection of pairs of concepts with similarity (and relatedness) for each pair indicated. The number of pairs included in each dataset is given in the figure beneath the datasets' names. In order to evaluate performance of our methods against NLP text-based methods, we can only consider pairs that are conjointly contained in the datasets across all levels 1–3. The next row in the figure (blue) shows the number of pairs conjointly contained in the human-rater as well the image datasets and the bottom row (black) shows the final minimal resulting intersection with the NLP datasets. These are the ones used by W2V-Neg and Glove. Clearly, only large enough intersections, all of which are given in the bottom row, could be used for our study. Intersection with only 3 or 9 pairs had to be ruled out.

The same procedure is performed on the right side of Fig. 9 B, but this time for concepts and super-categories on which we evaluate clustering consistency. As ground truth in this evaluation we take the division of 80 COCO concepts into 11 super-categories as done by humans. RCNN (2nd row) contains 79 of those, because the Mask R-CNN was not able to find the hair dryer. LabelMe overlaps with 76 concepts. The final overlap with the NLP datasets is given in the bottom row of panel B, which are those concepts with which we could perform the comparison.

A.5. Precision and recall

As an alternative to clustering consistency, the quality of the clustering can be assessed through a k nearest neighbor classifier. First, we explain how to create such a classifier. Consider the semantic vectors in Fig. 10, involving 12 concepts and three super-categories: Square, circle and triangle. We iterate over all super-categories (in Fig. 10 mid, square) and differentiate between element (square) and non-element (X). Using this binarization, we can decide between true-positive, false-positive, false-negative and true-negative from which precision and recall are computed:

$$\text{precision} = \frac{TP}{TP + FP}$$
$$\text{recall} = \frac{TP}{TP + FN}$$

Since this only involved one super-category, the procedure is repeated for every super-category and precision and recall scores are averaged, yielding the numbers reported in Fig. 11.

Results remain consistent in comparison to the ones obtained when using clustering consistency (compare to Fig. 4 from the main text).

Table 6
Scores for the hyper-parameter search of skip-gram. The number in bracket indicates the standard deviation over ten runs.

negative samples	epochs	dimension	clustering consistency	similarity	relatedness
10	50	128	0.423 (0.015)	0.438 (0.031)	0.397 (0.030)
25	50	128	0.413 (0.016)	0.455 (0.026)	0.433 (0.019)
50	50	128	0.404 (0.013)	0.452 (0.017)	0.441 (0.014)
25	10	128	0.228 (0.044)	0.265 (0.029)	0.258 (0.030)
25	100	128	0.468 (0.015)	0.474 (0.027)	0.441 (0.016)
25	50	64	0.478 (0.020)	0.472 (0.021)	0.451 (0.019)
25	50	256	0.372 (0.018)	0.403 (0.021)	0.386 (0.016)

A.6. Skip-gram hyper-parameter search

The Skip-Gram negative sampling training has multiple hyper-parameters. To enable a fair comparison with other methods, we investigate different choices for the number of negative samples being presented for every positive sample, the number of training epochs and the dimension of the concept vectors (see Table 6). Regarding the number of negative samples, we find the algorithm to be fairly stable and choose 25. The scores further suggest that ten epochs are too few while after 50 epochs the training seems to have converged in all cases and that even 64 dimensions already exhibit a good performance. Hence, we decided to run the training for 50 epochs with 64-dimensional concept vectors.

References

[1] Alejandro Agostini, Carme Torras, Florentin Wörgötter, Efficient interactive decision-making framework for robotic applications, Artif. Intell. 247 (2017) 187–212.
[2] Mohit Bansal, Kevin Gimpel, Karen Livescu, Tailoring continuous word representations for dependency parsing, in: Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2014, pp. 809–815.
[3] Roger J. Barlow, Statistics: A Guide to the Use of Statistical Methods in the Physical Sciences, vol. 29, John Wiley & Sons, 1989.
[4] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, Christian Janvin, A neural probabilistic language model, J. Mach. Learn. Res. 3 (2003) 1137–1155.
[5] David M. Blei, Andrew Y. Ng, Michael I. Jordan, Latent Dirichlet allocation, J. Mach. Learn. Res. 3 (2003) 993–1022.
[6] Elia Bruni, Nam-Khanh Tran, Marco Baroni, Multimodal distributional semantics, J. Artif. Intell. Res. 49 (2014) 1–47.
[7] Alexander Budanitsky, Graeme Hirst, Evaluating wordnet-based measures of lexical semantic relatedness, Comput. Linguist. 32 (1) (2006) 13–47.
[8] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, Pavel Kuksa, Natural language processing (almost) from scratch, J. Mach. Learn. Res. 12 (Aug 2011) 2493–2537.
[9] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, Cédric Bray, Visual categorization with bags of keypoints, in: Workshop on Statistical Learning in Computer Vision, ECCV, Prague, 2004.
[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, Li Fei-Fei, Imagenet: a large-scale hierarchical image database, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2009, pp. 248–255.
[11] Yansong Feng, Mirella Lapata, Visual information in semantic representation, in: The 2010 Annual Conference of the North American Chapter of the ACL, Association for Computational Linguistics, 2010, pp. 91–99.
[12] John R. Firth, A synopsis of linguistic theory, 1930–1955, in: Studies in Linguistic Analysis, 1957, pp. 1–32.
[13] Andrea Frome, Greg S. Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al., Devise: a deep visual-semantic embedding model, in: Advances in Neural Information Processing Systems, NIPS, 2013, pp. 2121–2129.
[14] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2014, pp. 580–587.
[15] Arthur M. Glenberg, David A. Robertson, Symbol grounding and meaning: a comparison of high-dimensional and embodied theories of meaning, J. Mem. Lang. 43 (3) (2000) 379–401, ISSN 0749596X, http://linkinghub.elsevier.com/retrieve/pii/S0749596X00927141.
[16] Stevan Harnad, The symbol grounding problem, Phys. D: Nonlinear Phenom. 42 (1–3) (1990) 335–346.
[17] Zellig S. Harris, Distributional structure, Word 10 (2–3) (1954) 146–162.
[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep residual learning for image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2016.
[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick, Mask r-cnn, in: IEEE international Conference on Computer Vision, ICCV, 2017.
[20] Felix Hill, Roi Reichart, Anna Korhonen, Simlex-999: evaluating semantic models with (genuine) similarity estimation, Comput. Linguist. 41 (4) (2015) 665–695.
[21] Douwe Kiela, Léon Bottou, Learning image embeddings using convolutional neural networks for improved multi-modal semantics, in: Conference on Empirical Methods in Natural Language Processing, EMNLP, 2015.
[22] Ákos Kádár, Afra Alishahi, Grzegorz Chrupała, Learning word meanings from images of natural scenes, Trait. Autom. Lang. Nat. Sci. Cognitives (Natural Language Processing and Cognitive Sciences (TAL)) 55 (3) (2014).
[23] Levy Omer, Yoav Goldberg, Neural word embedding as implicit matrix factorization, in: Advances in Neural Information Processing Systems, NIPS, 2014, pp. 2177–2185.
[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, C. Lawrence Zitnick, Microsoft coco: common objects in context, in: European Conference on Computer Vision, vol. 8693, ECCV, Springer International Publishing, ISBN 978-3-319-10601-4, 2014, pp. 740–755.
[25] David G. Lowe, Object recognition from local scale-invariant features, in: IEEE International Conference on Computer Vision, ICCV, 1999, pp. 1150–1157.
[26] Kevin Lund, Curt Burgess, Producing high-dimensional semantic spaces from lexical co-occurrence, Behav. Res. Methods Instrum. Comput. 28 (2) (1996) 203–208.
[27] Tomas Mikolov, Quoc V. Le, Ilya Sutskever, Exploiting similarities among languages for machine translation, arXiv preprint, arXiv:1309.4168, 2013.
[28] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, Jeff Dean, Distributed representations of words and phrases and their compositionality, in: Advances in Neural Information Processing Systems, NIPS, 2013, pp. 3111–3119.
[29] Jeffrey Pennington, Richard Socher, Christopher D. Manning, Glove: global vectors for word representation, in: Conference on Empirical Methods in Natural Language Processing, EMNLP, 2014, pp. 1532–1543.

[30] Joshua C. Peterson, Joshua T. Abbott, Thomas L. Griffiths, Adapting deep network features to capture psychological representations, in: Proceedings of the 38th Annual Conference of the Cognitive Science Society, 2016.
[31] Bryan C. Russell, Antonio Torralba, Kevin P. Murphy, William T. Freeman, LabelMe: a database and web-based tool for image annotation, Int. J. Comput. Vis. 77 (1–3) (2008) 157–173.
[32] Karen Simonyan, Andrew Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint, arXiv:1409.1556, 2014.
[33] Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, Christopher Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in: Conference on Empirical Methods in Natural Language Processing, EMNLP, 2013, pp. 1631–1642.
[34] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2015, pp. 1–9.
[35] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alexander A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, in: AAAI Conference on Artificial Intelligence, 2017, p. 12.
[36] Joseph Turian, Lev Ratinov, Yoshua Bengio, Word representations: a simple and general method for semi-supervised learning, in: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2010, pp. 384–394.
[37] Peter D. Turney, Patrick Pantel, From frequency to meaning: vector space models of semantics, J. Artif. Intell. Res. 37 (1) (2010) 141–188.
[38] Jason Weston, Samy Bengio, Nicolas Usunier, Large scale image annotation: learning to rank with joint word-image embeddings, Mach. Learn. (ISSN 1573-0565) 81 (1) (Oct 2010) 21–35.
[39] Caiming Xiong, Victor Zhong, Richard Socher, Dynamic coattention networks for question answering, in: International Conference on Learning Representations, ICLR, 2017.