Identifying Occluded Objects in Images Using Object Vector Representations *

Sailik Sengupta
Computer Science, Arizona State University
sailiks@asu.edu

Anagha Kulkarni
Computer Science, Arizona State University
akulka16@asu.edu

Elan Markov
Computer Science, Arizona State University
emarkov@asu.edu

Stephen McAleer
Mathematics, Arizona State University
smcaleer@asu.edu

Abstract

The goal of this project was to predict objects present in an occluded part of an image. This was accomplished by considering multiple pieces of that task: extraction of objects from images, natural language processing of image phrase description, and prediction of missing objects from given object data. Given an image, phrases that describe individual scenes within objects were extracted from human-generated captions on those images. From these phrases, the objects present in this phrases were obtained, using a natural language processing approach and a relationship graph approach. From the generated list of objects, a word to vector model was used for predicting a likelihood for the identity of the missing objects in the images. Our results show that, while the relationship mapping approach is more effective for prediction, the computational complexity of such an approach is substantial, which perhaps justifies a simpler NLP approach for processing a large number of images. Relative to human subjects, the word-to-vector model had modestly effective results under circumstances least favorable to human prediction. To the best of our knowledge, this is the first time such a task is being addressed. The results of this study recommend multiple approaches by which future studies can improve upon these results to obtain a more favorable performance in general.

1 Introduction

As humans, we are aware of the different relational semantics through perception that exist among different objects and environments. For example, parking lots consist of parked cars. This awareness, developed over the course of many years, helps us to learn and develop understanding about how different objects interact in the given environment. Using this learned information, we can effectively reason about what might be present in a place even if we cannot directly perceive it. For example, although we might only observe the closed drawer in a kitchen, we could reason that the drawer has knives or forks inside it based on our knowledge of what a kitchen drawer should contain. This brings us to the question, *can we achieve this kind of reasoning via the use of machine learning models?* In this project, we try to address this problem of the identification of objects in occluded parts of an image using machine learning models.

In the context of image processing, the rationale is that there is something in an image that you want to see, but you can't due to the present configuration of your visual sensors. The problem then is,

*This is the project report for CSE 575 Statistical Machine Learning, Arizona State University

how can you reason about the environment and guess what is there that you cannot see? A good solution to this problem will be significant in multiple areas for an artificially intelligent agent. As a motivating example, let us consider an autonomous vehicle approaching a 4-way crossing with no signal or stop signs. Let us say another car is coming to that crossing from an orthogonal direction, whose view is obstructed by a wall. Thus, given this knowledge (*known known*), the optimal decision for the automated car would be to drive at the same pace. If it was intelligent enough to predict that a car might be approaching from behind the wall, reasoning about the *known unknown* should suggest it to reduce the speed as it approaches the crossing.

In this project, we break this problem down into two general sub-problems, object data extraction and object prediction, and try to come up with solutions for each of them. The first part is to identify objects in an image and parse them in a way that will represent the semantic meaning of the image. To address this issue, we must identify local phrases in an image and obtain objects that are described within these phrases, which are ordered closer if they are obtained from the same phrase. For the latter part, we treat this order of objects obtained from an image as a sentence, where the objects represent words. We train a word-to-vector model with object data from images in the Visual Genome database and compare our prediction accuracy on this dataset using a $k-$fold validation approach.

The accuracy of our approach for test data appears somewhat poor at $(4 - 5\%)$. Apart from the fact that the problem is hard even for real humans (as we show in the experimental section), the use of existing methods (that are not 100% accurate) in the sub-problems results in a lower cascaded accuracy.

This work comes as a next step to existing work that identifies occlusion regions [1]. Although we plan to try an address some real-world problems by coupling them together, this is presently out of the scope of this project.

Previous work has focused on the use of geometric features–like shadow outlines, 3D spacial and depth information for figuring out what objects might be there in the occluded part of an image [2]. Naturally, partial information regarding missing objects are essential for this to work. In cases where such information is easily available, our method can use it to prune unlikely guesses, but even without them, our approach can work, whereas this fails. Another line of work considers capturing the environment through multiple viewpoints, to obtain partial information [3]. In many real world scenarios (like the autonomous car or closed drawer example), more viewpoints provide no extra information. Here, our approach can still provide meaningful object predictions.

2 Problem

In this sections we describe the problem at the high-level and the sub-problems of that general problem.

Given an image i, we have an area of the image that is obstructed. We assume that there a single object $\hat{o}$ in this obstructed region that we cannot see given i. The question is, *can we predict what* $\hat{o}$ *is?*

This problem is broken down into two sub-problems as stated above. The first sub-problem is that given an image i we need to represent it as a sequence of objects $\langle o_1, o_2, \ldots, o_n \rangle$ in the image where this sequence has some semantic meaning. Given an image in which one of the objects is obstructed from view, we can use the incomplete sequence of objects, $\langle o_1, \ldots, o_x, \ldots, o_n \rangle$, where $o_x = \hat{o}$, we now want to find $\hat{o}$.

Given we have a solution in place for the above sub-problem, we now need to predict the most likely object that $\hat{o}$ can be, given that the image has the other objects. Let the set of all objects in the images be V. Thus, essentially we want to find the probabilities $Pr(\hat{o} = v | o_1, \ldots, o_{x-1}, o_{x+1}, \ldots, o_n)$ for all the possible objects $v \in V$. The one with the highest value, say v_i is the most likely object that is occluded, i.e., $\hat{o} = v_i$. Since there can be multiple likely objects, we predict the top $k-$objects.

3 Methodology

In this section, we describe methods that provide meaningful solutions to the sub-problems mentioned above.

3.1 Extracting Objects from Images

The learning models used within this project involved object descriptions of the contents of an image. The Visual Genome dataset [4] was used for training and testing each model. Since this data was developed from phrase descriptions of scenes within an image, these phrases had to also be parsed into objects using a natural language processing parser. The details of this extraction process are described in the following sections.

3.1.1 Visual Genome Dataset

The Visual Genome dataset [4] is a library of over 100,000 images compiled at Stanford University. For each image, crowdsourcing was used in order to provide all images with a series of captions describing the contents of the images. This description was provided in the form of a rectangular box encapsulating the scene of interest, along with a phrase that explains the context of the encapsulated image. To be able to effectively use this data, the phrases data had to be extracted for each individual image and compiled into a text file. Then, this data could be converted into object data using an NLP parsing method.

This dataset was preferable due to the large size of the image database, the use of human-generated captions, and the use of rectangular boxes for labeling scene descriptions. While the images have a general, rather than domain-specific context, the images also contain groupings by which they can be separated into domain-specific data in order to run the analysis on a domain-specific subset of the data. While the set of potential objects within a domain-specific set of images is expected to be significantly reduced, the general pattern-matching that is developed from a general understanding of many scenes is also important to the training of the model, so both the general and domain-specific cases are of interest for these models. Human-generated captions reduce the cascading error that could be expected from using a dense captioning tool; while further studies could investigate the relative effectiveness of dense captioning toolboxes relative to human-generated captions, in this case the higher accuracy of human-generated data was desired. As the spatial proximity of objects played a critical role in the training of each model, a simple format for the spatial location and size of scenes was preferred to a more specific, but more difficult to specify, segment description of location. For these reasons, the Visual Genome [4] image dataset was the dataset selected for training and testing the models within this study.

An API was provided by the Visual Genome site for processing images from the online database of images [4]. From this database, the phrases could be extracted for processing. Each image contains an average of approximately 200 scene descriptions; the form of the output is given below. Note that the format of this output lists the x-coordinate (from the top left corner), y-coordinate (from the top left corner), object width, object height, and the phrase, respectively.

```
421 57 82 139 the clock is green in colour
194 372 182 109 shade is along the street
```

While the caption data provided is in the form of phrase descriptions, these phrase descriptions need to be processed into object descriptions in order to be used by the prediction models. The Visual Genome API provided NLP functionality in order to extract objects; this was performed by mapping relationships between words within phrases to other words within the same phrase and to other words within different phrases. For example, for the phrase ”the clock is green in colour” above, ”clock” and ”green” would be associated in that ”green” is a description of ”clock” and ”colour” would be a description of ”green.” However, the issue with constructing such a relation mapping is time complexity; a sample of 100 images was processed using the API’s relation mapping toolbox, and this task was completed in approximately 10 hours, or 6 minutes per image.

In order to obtain the object descriptions, the processing method using relation mappings as provided by the Visual Genome API was determined to be too slow to be effective. As such, it was determined that an NLP processing step was necessary in order to extract the objects. In addition,

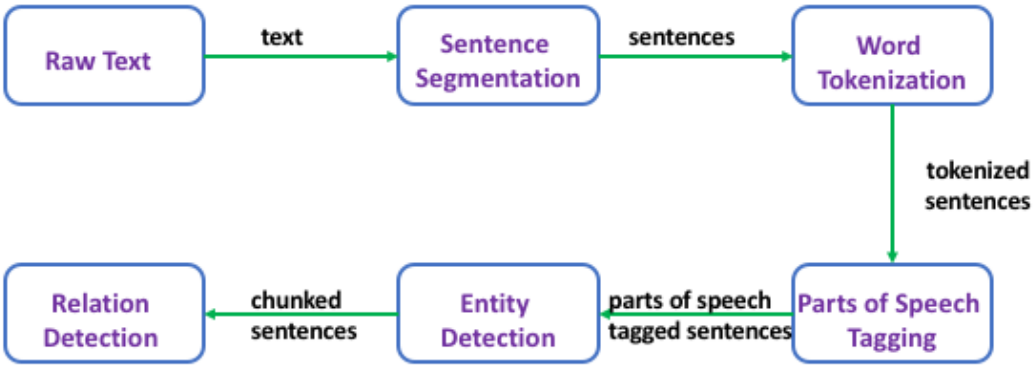

Figure 1: Information extraction architecture [5]

a JSON file containing the output of the relation mapping for each image was available; this was used alongside the NLP-generated object data in order to compare the effectiveness of each method. While this object data was provided in the case of this set of images, further classification would require additional image data, which would be available in the phrase form only. Since it could not be expected that the relationship mappings would be made available from a previous processing of that data, a separate method for extracting object data was necessary, motivating the use of an NLP preprocessor designed independently of the Visual Genome relation mapping toolbox.

A sample of two output objects is given below, with a similar format as before:

```
119 338 274 192 shade
77 328 714 262 street
```

The details of the NLP preprocessor for parsing phrase data are given in the following section.

3.1.2 Parsing using Natural Language Processing

The phrases provided by Visual Genome dataset [4] for images, were used to extract the objects present in the images. To this end, we performed several NLP techniques on the given phrases. We performed the following three steps to extract the objects from the phrases:

- The first step was to extract nouns from the phrases representing the objects in the images.
- The second step was to remove the synonyms of the objects obtained from the previous step.
- The third step was to remove multiple references to the same object to get unique references to each object in the image.

We will now explain each of these steps in detail.

Step 1: Extracting Nouns from Phrases

In this step, for each image, we process the available phrases to extract nouns from them. The process resembles the procedure illustrated in Figure 1. This process starts with segmenting the text document into sentences (or lists of strings). Each sentence is further segmented into words using the tokenizer. It converts the sentences into segments or tokens, that resemble words or group of words, depending on the words in the sentence. These tokens are further tagged using a parts of speech tagger.

Parts of speech tagging process classifies the tokens obtained in previous step into their corresponding parts of speech. Since we are interested in extracting objects from the phrases, we obtain the tokens classified with noun-form tags. The possible noun-forms tags are `NN`, `NNS`, `NNP`, `NNPS`. These tags are assigned to the tokens using either unigram or n-gram taggers. Unigram taggers [7]

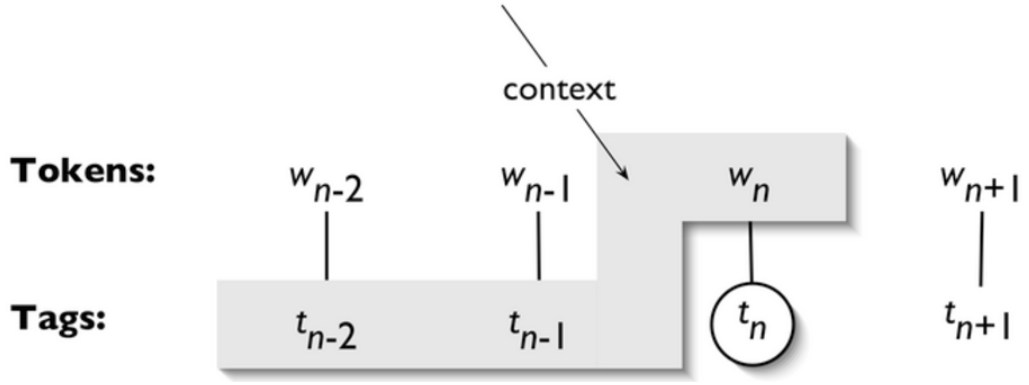

Figure 2: N-gram tagger context [6]

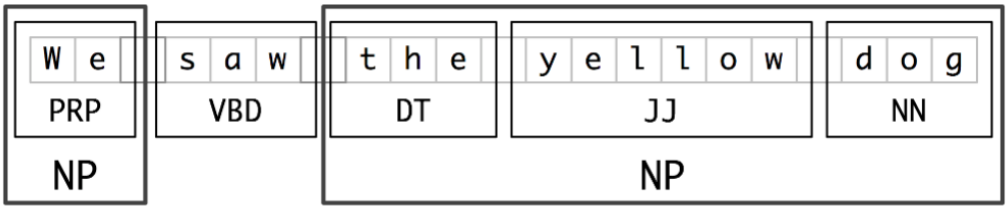

Figure 3: Parts of speech tagging at token and chunk levels [6]

assign the tags naively to the tokens, by just considering the most likely tag that the word usually is assigned. N-gram tagger [7], on the other hand, considers context of the previous token tags in deciding the tag of the current word. As shown in Figure 2, the n^{th} token is the context and $n-1$ previous token tags are used in assigning the most likely tag for the n^{th} token. For example, consider the word "address": it can take noun form in the sentence, "what is your address?", but in another sentence, "can you address this issue?", it takes the verb form because it is preceded by the pronoun "your". A bigram tagger (n-gram with n equal to 2) would be able to capture the actual tag of the word address in the example sentence. These n-gram taggers are used for tagging the tokens as shown in Figure 3. Further processing can be done on the tags to extract entities, this is called noun phrase chunking [6]. This segments the sentences in chunks of noun phrases with multiple tokens as outlined by the bigger boxes in Figure 3. For our purposes of extracting objects from the phrases, we do not use the entities, but rather stop the extraction process at token level.

Step 2: Removal of Synonymous Nouns

The phrases extracted from the Visual Genome dataset [4] were labeled by users and had the object references were riddled with synonyms. In order to tackle this issue, we made use of `wordnet synsets` [8] corpus. This corpus provides help with deriving conceptual relationships between objects through its sets of synonyms, hyponyms, hypernyms and antonyms. We removed all the synonymous references to the objects. For example, an image with two men had synonymous reference to words "person" and "guy", apart from the word "man". In this step, all such references were removed to further converge the list of objects for each image.

Step 3: Removal of Multiple References to an Object

The phrases extracted from the Visual Genome dataset [4], had multiple references to same objects. This occurred because there were phrases talking about an object's relationship with various other objects. For example, an image with a man was described by phrases like, "the man is wearing a red shirt", "the man is wearing sneakers", this resulted in multiple references to the same object man. In order, to deal with this issue, we used the spatial location of the objects referenced in the

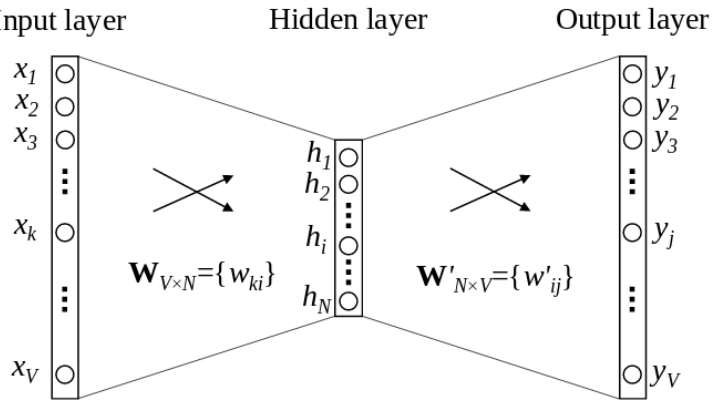

Figure 4: One-input Continuous Bag of Words model [9]

phrases to decipher whether the objects were unique or not. To this end, we used the bounding boxes around the objects to see whether there was any overlap between the boxes, and accordingly decided whether the references to the objects were unique or not. Thus we removed multiple references to an object.

After the preprocessing step, the objects lists for images were trained on the word-to-vector model. The details of this are presented in the following sections.

3.2 Predicting likely objects from incomplete object sequences

We essentially want to calculate the probability $Pr(\hat{o} = v|o_1, \ldots, o_{x-1}, o_{x+1}, \ldots, o_n)$ for all $v \in V$ where o_x is the missing object. To do this, we use word-to-vector.

3.2.1 Word to Vector

We train a word-to-vector model using the object sequence data we obtain from Visual Genome [4] above. Once trained, we use the model to obtain the required probability values. We provide a brief introduction on word-to-vector models, emphasizing on the Continuous Bag Of Words (CBOW) architecture and hierarchical sampling that we use in this project.

One-input CBOW

Before we describe the CBOW we used for our project, that comprises of multiple input objects (in context) to predict a missing object, we introduce a simple one-input CBOW a architecture, that will help the reader grasp the technical concepts that follow.

An architecture for this is shown in Figure 4. We assume the size of our vocabulary, which is the number of unique objects that can be present in an image, is V. The hidden layer has a size of N (neurons). An input word is a *one-hot encoding*. Given a word, only the x_i representing this word will be one, where all other $x_j = 0$ $(\forall\, j \neq i)$.

As per the figure, we will have to train 2 sets of weights– input to hidden layer ($W_{V\times N}$) and hidden to output layer ($W'_{N\times V}$)– for obtain a good prediction, which is essentially a *one-hot encoding* of the predicted word (or object in our case).

Given an input word w_k (i.e., $x_k = 1$ and $x_j = 0\, \forall j \neq k$), we are just copying the $k-$th row of the weight matrix W to the hidden layer,

$$h = W^T x = v^T_{w_k}$$

where v_{w_k} is the vector representation of the input word w_k.

Using the weights between the hidden and the output later (W'), we can now determine a score u_j for each word w_j in the vocabulary,

$$u_j = v'^T_{w_j} h$$

where v'_{w_j} is the $j-$th column of W'. Notice that the word-to-vector does not use a sigmoid activation function [10]. To convert this score into probability values that provides the likelihood of each output word given the input word, we can use a softmax log-linear classification function as follows,

$$\begin{aligned} Pr(w_j|w_k) &= \frac{\exp(u_j)}{\sum_{p=1}^{V}\exp(u_p)} \\ &= \frac{\exp(v'^T_{w_j}h)}{\sum_{p=1}^{V}\exp(v'^T_{w_p}h)} \\ &= \frac{\exp(v'^T_{w_j}v^T_{w_k})}{\sum_{p=1}^{V}\exp(v'^T_{w_p}v^T_{w_k})} \end{aligned}$$

Now all we need to do is define our loss function and train the weights of this architecture by the backpropogation algorithm. We use a logarithmic loss function as follow,

$$\begin{aligned} L &= -\log y^* \\ &= Pr(w_{j^*}|w_k) \\ &= u_{j^*} - \log\sum_{p=1}^{V}\exp(u_p) \end{aligned}$$

where w_{j^*} is the actual output word in the training examples. Now for each word u_j,

$$\frac{\delta L}{\delta u_j} = y_j - t_j = e_j$$

where $t_j = \mathcal{I}(j = j^*)$ is the indicator function. Thus, e_j represents the error in prediction. If the network predicts correctly, the output would have $y_{j^*} = 1$ and $y_j = 0$ ($\forall\ j \neq j^*$), which implies $e_j = 0$. Otherwise, there will be negative error values of j^* and positive error values for j ($\neq j^*$).

We now mention the updates that can be used to update each of the weights in between the hidden and output layers of the neural network. The gradient can be calculated as follows,

$$\frac{\delta L}{\delta w'_{ij}} = \frac{\delta L}{\delta u_j}.\frac{\delta u_j}{\delta w'_{ij}} = e_j.h_i$$

Hence, the Stocastic Gradient Descent (SGD) update on the parameters would be,

$$w'^{new}_{ij} = w'^{old}_{ij} - \eta.e_j.h_i$$

or,

$$v'^{new}_{w_j} = v'^{old}_{w_j} - \eta.e_j.h_i$$

where $\eta > 0$ is the learning rate. Lower the error for the $j-$th word, lower the requirement for updating the corresponding weight column in the weight matrix W'.

We can now backpropogate the error to update the weight matrix W. For this purpose we first calculate,

$$\frac{\delta L}{\delta h_i} = \sum_{j=1}^{V}\frac{\delta L}{\delta u_j}.\frac{\delta u_j}{\delta h_i} = \sum_{j=1}^{V} e_j.w'_{ij} = e_{h_i}$$

where e_{h_i} is the error resulting because of the hidden layer node h_i. Since $h_i = \sum_{k=1}^{V} x_k.w_{ki}$, we can calculate the gradients for the weights in between the input and hidden layer as follows,

$$\frac{\delta L}{\delta w_{ki}} = \frac{\delta L}{\delta h_i}.\frac{\delta h_i}{\delta w_{ki}} = e_{h_i}.x_k$$

Notice that $x_k = 1$ for only the one input word k and zero for all the rest ($x_j = 0\ \forall\ j \neq k$). Hence, the only weight that will be updated is,

$$v^{new}_{w_k} = v^{old}_{w_k} - \eta.e_{\vec{h}}$$

where $\frac{\delta L}{\delta W} = x \otimes e_{\vec{h}}$ is the derivative of the entire weight matrix represented as the tensor product between input vector x and error vector representing error for all the hidden nodes ($e_{\vec{h}}$).

We are now in a position to talk about the multi-word context that we use for our project.

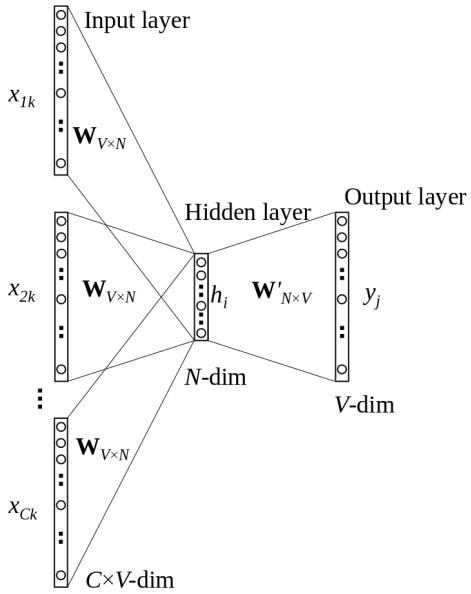

Figure 5: Continuous Bag of Words model [9]

Multiple input CBOW

To add more context on how we use this architecture for our project, we will use a running example. In this example, the module mentioned in subsection 3.1 generates the sentence *"guitar, hand, y_j, hat"* where we are interested in knowing what y_j is.

As seen in Figure 5, the work to vector is essentially a neural network architecture, with C input words ($x_{ik}\ \forall\ i\ \in 1, 2, \ldots, k$) that represent the context and a single output word y_j that represents the predicted word. Here, C refers to the $2 \times window\_size$, which essentially considers how much context we should take into account for generating a good prediction for y_j. For example, if we have $window\_size = 1$, then $C = 2$. The inputs x_{1k} and x_{2k} would be the one-hot vector representations for the words *hand* and *hat* in our example, encompassing a context of 1 word on both sides of the missing word (y_j) we want to predict.

The values of the hidden layer nodes have now become an average vector output of the C input word vectors,

$$\begin{aligned}\vec{h} &= \frac{1}{C}W^T(x_1 + x_2 + \ldots + x_c) \\ &= \frac{1}{C}(v_{w_1} + v_{w_2} + \ldots + v_{w_c})^T\end{aligned}$$

where v_{w_i} is the input vector of the work w_i. The loss function is,

$$\begin{aligned}E &= -\log Pr(w_O|w_{I1}, \ldots, w_{IC}) \\ &= -u_{j^*} + \log \sum_{j'=1}^{V} \exp(u_{j'}) \\ &= -v'^T_{w_O}.\vec{h} + \log \sum_{j'=1}^{V} \exp(v'^T_{w_j}.\vec{h})\end{aligned}$$

Notice that the loss function now tries to maximize the log likelihood of the output word given the context of work it has. In our case, given that it can see the objects *hand, guitar* and *hat*, it will try to choose an object that maximize the log likelihood or in turn, minimize the loss function.

The update equations for the weights of the neural net for our case are thus,

$$v'^{new}_{w_j} = v'^{old}_{w_j} - \eta.e_j.\vec{h}\ \forall\ j = 1, 2, \ldots, V \tag{1}$$

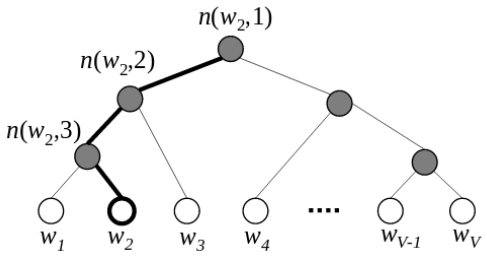

Figure 6: Hierarchical Sampling tree: The white units represent the works in the vocabulary. The path to word w_2 is shown. $L(w_2) = 4$ and $n(w, j)$ represents the $j-$th node on the path from root to the target word w [9].

and,

$$v_{w_{Ic}}^{new} = v_{w_{Ic}}^{old} - \frac{1}{C}.\eta.e_{\vec{h}} \; \forall \; c = 1, 2, \ldots, C$$

Notice that in this case we will land up updating all the weights that are related to any of the input word in the context.

Computational Efficiency

If a reader notices carefully, we have two vectors for each word– and input vector v_w and and output vector v'_w. Although updating v_w requires you to care about only the words that are present in the context of a training example, updating v'_w needs you to consider the vectors for all the words in the vocabulary (as is evident from Equation 1). To reduce the computational inefficiency for the latter part, we use Hierarchical Sampling technique.

We provide the main intuition behind this technique and let the inquisitive reader figure out the technical details [11, 12]. In this architecture, each of the words are represented as nodes of a complete binary tree where the nodes in the last layer represent the words in the vocabulary (as shown in Figure 6).

Once we have this tree, the output word does not need an output vector representation. Although a vector representation for the $V - 1$ hidden nodes are needed. So how does this help?

Without diving too much into the technical details, we provide you the loss function for a single input case,

$$E = -\log Pr(w = w_O|w_I) = -\sum_{j=1}^{L(w)-1} \log \sigma([\![n(w, j+1) = ch(n(w, j))]\!]) \; v_j'^T.\vec{h}$$

Thus, for updating the weights in the output layer, the partial derivative is calculated only for the inner vectors of the actual output word and not all the inner vectors. Thus, the update equation now updates only $\log |V|$ vectors (on the path from the root to the actual output word) and not $|V|$ vectors, effectively reducing the computational inefficiency. For more details on any of the topics mentioned here, we refer the reader to [9], from which our write-up draws inspiration. (The notations are kept the same to help the reader quickly grasp the original material.)

4 Experimental Results

In this section, we first talk about how difficult the problem is for a human to address. We then talk about the accuracy of our solution to this problem.

4.1 Human Experiments

While the performance of the machine is of interest in the design of these algorithms, it is also important to note that humans would not be expected to have perfect or near-perfect accuracy in

the prediction of images due to the difficulty of the problem of identifying such missing images. As such, it was desirable to obtain data about the performance of humans in the identification of missing objects to serve as a baseline for judging the effectiveness of the model. The methodology and results of the study are detailed in the following sections.

4.1.1 Human Test Methodology

In order to simulate missing objects, several images were provided in which a rectangular box covered up specific objects within the image. The human subjects were then asked to guess the object that is covered by the box. For each individual object, the human subjects were given five guesses as to the object's identity; a correct prediction was considered to be one in which the object contained within the box was one of the guesses provided. Each of the test subjects, when asked, agreed that they would not have had any further guesses if more than five guesses were available per image; this suggests that the human model of solving this problem converges upon a few likely objects and will not develop any predictions beyond those deemed most likely. Each image had three occluded objects: a red-boxed object, a yellow-boxed object, and a green-boxed object, based on the relative difficulty of the prediction. The image in Figure 7 shows an example of how an image is modified to be shown to a human subject with certain objects occluded; the meaning of the boxing colors is described below.

- A red-boxed object is one that is deemed to be difficult for a human to predict; there is little context provided as to the contents of the image and a human must guess from the general context of the entire image. The accuracy of human guesses on this set of objects is expected to be relatively low. An example of such an object would be a cell phone on a desk; while a human would not be surprised by a cell phone being on top of a desk, the context of an object being on the desk is not enough to suggest that said object is a cell phone with any reasonable degree of accuracy.
- A yellow-boxed object is one whose identity is not immediately obvious, but whose identity could be deduced through reasoning about the image and its contents. The accuracy of human guesses on this set of objects is expected to be moderately high. An example of such an image would be a tennis ball when it is seen that a person is looking in the direction of that missing object with a racket in a position that suggests that he or she is about to strike a ball.
- A green-boxed object is one for which a human should have little difficulty guessing, in that the context of the image should make it immediately apparent what is contained within the image. The accuracy of human guesses on this set of objects is expected to be very high. An example of such an object would be a headlight at the left or right corner of the front of a car; based on a human perception of a car design, it is immediately obvious that the correct object to be contained in that location is a headlight.

Each of these images, with the proper objects blocked out, were provided in the form of a Google Forms survey to ten human subjects. Their results were compared to the actual objects contained behind the boxes. The results of this study are tabulated in the following section.

4.1.2 Human Results

The following table details the raw results of the human study.

ID	Red Ratio	Yellow Ratio	Green Ratio	Average Ratio
61533	0.1	0.6	1	0.567
61585	0.7	1	0.7	0.8
61607	0	0.7	1	0.567
713273	0.7	0.9	0.9	0.833
713347	0.4	0.4	1	0.6
150382	0.2	0.2	0.5	0.3
150427	0.778	0.444	0.889	0.704
712987	0	0.889	0.889	0.592
498341	0.111	0.556	1	0.555555556
713202	0.111	0.889	0.778	0.592
All	0.3125	0.656	0.865	0.611

Figure 7: Comparison of full image (left) and with boxed occluded objects (right). Visual Genome database [4], image number 61533.

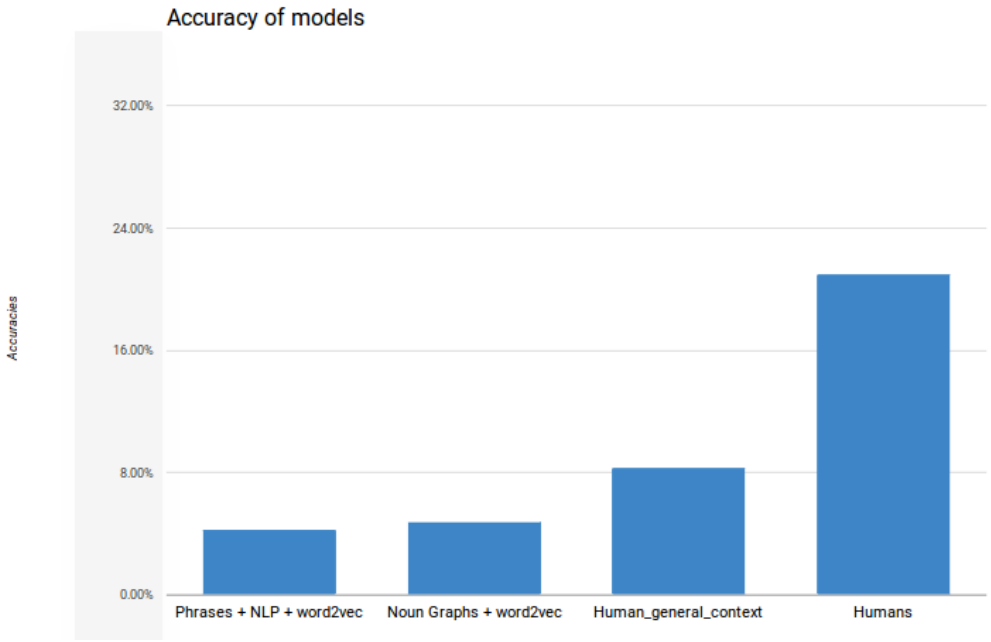

Figure 8: Results comparing our approach to Naive Bayes and Human experiment data.

As expected, the accuracy for predictions of the red-boxed data is lower than the accuracy for the yellow-boxed data, which is smaller than the accuracy for the green-boxed data. However, the results for the red images were deemed to be somewhat questionable for certain cases due to functionality not implemented within the learning models which a human subject would be able to use. For images 61585 and 713273, the correct answer was "nothing," an answer which the model could not predict since "nothing" is not an object which is marked within the object list. This reduces the total red ratio to 21.05%. Furthermore, images 713347, 150382, and 498341 were determined to have a degree of local context more suitable to a yellow-boxed images; the removal of the five images listed above will reduce the red ratio to 8.33%. This shows that, within only the general context of the image as a whole, the accuracy of human predictions is significantly reduced relative to the situation of if there is some specific, local context to draw upon. These values will be used to compare to the prediction models studied.

4.2 Our approach

We cannot compare our approach to any existing work since none exist. Hence, we comparison our results to how humans did in Figure 8.

For accuracy values plotted are results of of a $k-$fold validation with $k = 10$. We obtained accuracy results with $k = 5, 6, \ldots, 12$ and $k = 10$ yielded the best results.

For the word to vector approach, we used a prediction set size of 5%. The $window\_size = 2$ yielded the best results. For most images, given ($C =$) 4 objects around it, we could identify the object with sufficiently good accuracy. If we considered fewer objects in the context, there could be a lot of objects that could be predicted, reducing the accuracy. If we consider more objects, the object predictions particularized to images and general prediction accuracy becomes low.

The accuracy values for our approach is sufficiently low (approx. $4 - 5\%$). There are two major reasons for this low accuracy. First, the task is hard when contextual information is not sufficient to infer the correct object, even for humans (see $human\_general\_context$ column). Second, our approach relies on generation of phrases from images, Natural Language Parsing of phrases to infer objects and lastly a word-to-vector model for prediction. Each of these tasks have an accuracy at least somewhat below 100%. Cascading them to figure out a solution to such a difficult problem can only help us to an extent.

One way of solving the former problem we mention would be to consider images only specific to a certain domain. To test this hypothesis, we took all the images in the Visual Genome dataset [4] where a skiing scene was being portrayed. The selection was based on the fact that this topic had 3737 images, which was the largest domain-specific subset of the images. For this scenario, the accuracy was only 2.2%. But this time, although the problem was much easier, our approach that is data intensive, failed to work well. This suggests that the number of images adds significantly more to the predictive power than the domain-specificity of the images.

5 Discussion and Future Work

Although we have cited reasons for the low accuracy for our approach, an interesting question would be, *"How does one address these weaknesses?"*. In this section, we talk about some information humans use in order to address this task. Using this, we provide some insights on how we can design a better approach.

5.1 Human Results Analysis

One factor that the human study reveals is a general pattern-matching ability found in humans which is not seen in the algorithmic approach to making predictions. One notable example is that in Figure 7, the identity of the yellow object would be instantaneously apparent to a human if they notice the shade that takes the shape of a ball in the image. While this object is spatially removed from the ball itself, it ends up being relevant to the prediction of the missing object, showing the weakness of an approach which only considers the local context of the surroundings of the missing object. Furthermore, such a general pattern-matching approach is not domain-specific; the context of shade giving clues as to the identity of objects is not relevant only in the context of a tennis court, but in the general context of all images in which there is a visible light source. This pattern matching involves noticing patterns in multiple, highly distinct scenarios, which a human can deduce well but which an algorithmic approach which attempts to classify objects will have difficulty recognizing.

The human model of prediction is based on their own experiences over many years over a wide range of images in a wide range of contexts, which provides a means by which a human could recognize these patterns. This large degree of interconnected concepts between distinct image types is very useful for deducing the context of images, although it does require a very large amount of "training data" to be effective. This is one likely explanation for why the full dataset, despite adding a substantially larger vocabulary of potential objects to the set of possible guesses, gives worse results due to the decrease in the number of training images. While some additional accuracy could likely be obtained from a domain-specific dataset, the human pattern-matching and deduction approaches to

object prediction suggest that the size of training data, more so than domain-specificity, determines the effectiveness of the prediction model.

5.2 Algorithmic Approaches to Improvement

Based on the results of the word-to-vec algorithm and the human results, a few approaches to further improve accuracy of the prediction task is suggested.

One avenue that can lead to better results would be to better incorporate global image context into the model. A similar approach to ours might involve ranking all objects detected in the image and assigning a distance for each one. This might provide slightly better results than our approach, since it more accurately assesses closeness rather than treating all objects in a given window as a bag of words. For instance, in the word-to-vec approach, objects are be ranked in order of distance to other objects based on spatial proximity. When humans reason about what an occluded object might be, a lot of relevant information is information about objects that are spatially close to that object. However, as seen in the shade of the tennis ball in Figure 7, some important context information may be substantially spatially removed from the image yet still be relevant to the prediction. An ability to include some of this more global information would help in improving the overall prediction rate by allowing the algorithm to infer some of this context.

Another approach is to use clustering techniques for bucketing groups of words into semantically-similar ideas. For instance, the words ”horse,” ”mule,” and ”donkey” refer to very similar objects, so if we treat instances of donkeys and mules as horses, the model will have an easier time guessing occluded classes of objects, which will help offset the negative effect of the sparsity of data. This clustering can be done in the Word2Vec latent space trained on a large corpus of writing using a clustering technique such as k-means. If trying to predict a class of closely related objects, rather than a highly specific one, it is likely that the effectiveness of the predictions will improve.

A further approach would be to include additional information about the area surrounding the region to be inferred. For instance, the color and shape of materials surrounding the occluded region could help by predicting which objects occur most given those colors and textures. To best accomplish this task, we would need to employ a convolutional neural network trained on identifying color or texture and then incorporate the latent representation of the surrounding region into the Word2Vec model.

6 Conclusion

In this project we proposed the problem of inferring missing objects in an image based on the identity, size, and spatial location of other objects in the image. To address this problem, we broke the problem up into several sub-problems addressed separately: phrase and object extraction from image data, NLP parsing of the data, and prediction of missing objects from the remaining object data.

The Visual Genome image database [4] provided over 100,000 images from a general context from which human-generated phrase descriptions of the image could be extracted. To convert these phrase descriptions to objects, two methods were used: an NLP parser and Visual Genome’s relationship mapping approach. While the relationship mapping proved to have somewhat better results in the classification stage, this method involved substantially more computational time, and these object descriptions could only be obtained from a previous run of the relation mapping tool by the Visual Genome creators. This suggests that for future processing of data, the NLP preprocessing approach is likely to be more effective due to a relatively modest accuracy loss with a substantial gain in computational performance.

We then researched on existing Natural Language Processing techniques to obtain objects present in the image from these phrases. To this end, we used n-gram parts of speech tagging on tokenized sentences to extract the nouns (objects) from the phrases. The objects so obtained were then further filtered to remove any synonymous or repetitive references to a unique object using `wordnet` corpus. While this approach did yield an object list that was less effective for predicting missing objects than the relation mapping approach, this preprocessing approach was completed much more quickly with a relatively modest decrease in results.

For the problem of predicting missing objects, random objects were removed from an object list, and a prediction model was used to predict the identity of that missing object from surrounding objects. To solve this, we trained a Word-to-Vector model using a Continuous Bag Of Words (CBOW) model and used it to predict the missing object in the image, given the context of other objects in the image. To bypass the computational inefficiency of the word-to-vector model as a result of a very large number of objects, a hierarchical sampling approach was used for training our word-to-vector model, which substantially improved the runtime of the model. Under the most unfavorable conditions (general context only, no objects containing "nothing"), the performance of the human was only modestly better than that of the word-to-vector model. However, the human experiments revealed numerous qualities of a human prediction approach which the current algorithm is not capable of using, suggesting multiple paths by which a more refined algorithm could be used for improving performance in the future.

In future studies of image prediction, it is recommended that an image prediction algorithm make use of a deductive pattern-matching approach to better infer the context of an image for prediction. While this approach will necessarily involve a very large amount of training data, this approach has shown that the effectiveness of predictions is largely dependent upon the quantity of data available for training, more so than on domain-specificity or on reducing the object vocabulary. A general context for image prediction provides much data by which such classifications can be improved.

References

[1] Ahmad Humayun, Oisin Mac Aodha, and Gabriel J Brostow. Learning to find occlusion regions. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2161–2168. IEEE, 2011.

[2] Yu Xiang and Silvio Savarese. Object detection by 3d aspectlets and occlusion reasoning. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 530–537, 2013.

[3] Smitha Suresh, K Chitra, and P Deepak. A survey on occlusion detection. In *International Journal of Engineering Research and Technology*, volume 2. ESRSA Publications, 2013.

[4] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *arXiv preprint arXiv:1602.07332*, 2016.

[5] Steven Bird, Ewan Klein, and Edward Loper. Natural language processing with python. chapter 7: Nltk: Extracting information from text, 2009.

[6] Steven Bird, Ewan Klein, and Edward Loper. Natural language processing with python. chapter 5: Nltk: Categorizing and tagging words, 2009.

[7] Helmut Schmid. Probabilistic part-ofispeech tagging using decision trees. In *New methods in language processing*, page 154. Routledge, 2013.

[8] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[9] Xin Rong. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014.

[10] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[11] Andriy Mnih and Geoffrey E Hinton. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088, 2009.

[12] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252. Citeseer, 2005.