Chapter 6

Automatic image description: a first impression

6.1 Introduction

The field of automatic image description lies at the intersection of Natural Language Processing and Computer Vision (Bernardi et al., 2016). It is closely related to the field of Natural Language Generation (NLG, see Gatt and Krahmer 2018 for an overview). Researchers in this area aim to produce systems which can automatically generate understandable text, either on the basis of data (e.g. stock market figures, patient data, images), or on the basis of another text (e.g. for the purpose of summarization or text simplification). Generally speaking, there are two approaches to build NLG systems: (1) writing rules and templates that specify what the generated text should look like; and (2) training a system to learn the correct behavior from example data. We will ignore planning-based approaches (see the discussion in Gatt and Krahmer 2018), because we are not aware of any planning-based image description systems. Although early work in automatic image description used a rule-based approach (e.g. Kulkarni et al. 2011; Mitchell et al. 2012; Elliott and Keller 2013), most recent work takes a more data-driven approach (e.g. Vinyals et al. 2015; Xu et al. 2015; Wu et al. 2017a; Dai et al. 2017). Hence, in this chapter, we will focus on the latter.

6.1.1 Goal of this chapter

This chapter serves as an introduction to the second part of this thesis, and presents an overview of the components used in current image description systems. Besides introducing important terms and concepts from the literature, we will also provide an error analysis of a data-driven automatic image description system (Xu et al., 2015). This will give us an indication of the quality of state-of-the-art image description technology.

6.1.2 Structure

This chapter consists of two parts. Following this introduction, we first cover the basics of neural networks, and then discuss the typical components of neural image description systems (CNNs in §6.3 and RNNs in §6.4). Finally, we will describe Generative Adversarial Networks (§6.5). Our aim here is to give the reader a basic understanding of how neural architectures work, without diving into their formal definitions. We conclude this part with a section on possible future improvements (§6.6).

The second part of this chapter (§6.7-6.11) focuses on error analysis, a practice used to understand the strengths and weaknesses of individual systems. We present a taxonomy of errors made by Xu et al.’s (2015) system, and annotate those errors to get a sense of their distribution. Through this annotation effort, we show where there is still room for improvement for this system (and, by extension, systems with a similar architecture).

6.1.3 Sources

This chapter draws from several overviews of the field, in particular:
Chapter 6 Automatic image description: a first impression


The second half of this chapter is based on the following work:


6.2 Neural networks

Neural networks are machine learning models that are loosely inspired by the human brain. They consist of artificial neurons, that are usually connected to each other in layers (groups of neurons). Figure 6.1 shows a schematic of an ‘actual’ neuron. It consists of a cell body that receives input through its dendrites. If the signal is strong enough to surpass a threshold value, the cell fires a signal through the axon to the axon terminals, which pass the signal through to other cells. Figure 6.2 shows an artificial neuron. The input nodes \((x_1...n)\) are attached to the neuron through weighted connections. The neuron takes the sum of the inputs multiplied by their weights, and transforms the result through some predefined function: \(f(\sum^n_{i=1} x_i \cdot w_i)\). When fed with example \{input, output\} pairs, neural networks are programmed to learn a mapping between the input and the output, by modifying the weights on their connections by back-propagation (Rumelhart et al., 1986). This supervised learning process is referred to as training.

Figure 6.1 A neuron. Original image by edgato on Openclipart.org (public domain).

Figure 6.2 Artificial neuron.

To illustrate the training process, let us suppose that the neuron just computes the identity function \(f(x) = x\). Further suppose that the task of the network is to predict whether a number is even or odd, and that numbers are fed to the neuron in binary form. E.g. 1 is represented as \([0, 0, 0, 1]\), four is represented as \([0, 1, 0, 0]\), and nine is represented as
For odd, the network should output 1, and for even, the network should output 0. Finally, assume that weights are initialized as random floats. With an initialization of \[ w_{1,2,3,4} = [0.1, 0.4, 0.3, 0.9], \] an input of \([1,0,0,1] \) would yield 1.0 (the sum of all inputs multiplied by their weights). Accidental success! But for other numbers, this initialization would not give the right result. For example, with an input of 6 ([0, 1, 1, 0]) the neuron would yield 0.7. During training, the weights that contributed to the error are adjusted to generate a better result in the future. Eventually, the weights for this example should end up as \([0, 0, 0, 1.0], \) because only the final digit of the binary number provides relevant information about whether the number is odd or even.

A neural network is simply a collection of neurons that are all connected together as a directed acyclic graph. Figure 6.3 shows an example of such a network, with the neurons organized in layers. The input layer feeds into a hidden layer, which is connected to another hidden layer, which feeds into the output layer. The hidden layers are called ‘hidden’ because they are not as directly accessible in the same way as the input or the output. Having multiple of these layers is useful, because they allow for more complex transformations of the input data, which in turn allows us to solve more complex problems. An intriguing property of neural networks is that there is no symbolic representation of ‘what is learned.’ Knowledge of how to solve the problem is stored holistically as connection weights.

![Figure 6.3 A neural network with two hidden layers.](image)

### 6.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs, LeCun et al. 1998) are commonly used for Computer Vision tasks, such as optical character recognition (OCR) and image labeling (Russakovsky et al., 2015). Rather than taking a one-dimensional vector as their input (like most neural networks), CNNs operate over two-dimensional grids or matrices. This is useful for image processing, because digital images can be represented as matrices with pixel values. Figure 6.4 shows an example, using a picture of the number four.

Let’s say we want to build a system that automatically recognizes handwritten numbers, by correlating different visual features with the desired output. The image on the left of Figure 6.4 shows three relevant features (according to human intuition), highlighted in blue:

---

1The position of the digits in a binary number correspond to powers of 2, starting from \(2^0\) in the rightmost position. To obtain the value of a binary number, multiply the value in each position with the corresponding power of two, and sum the results. Hence \([1, 0, 1, 1] = (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) = 11. Because 2^{1\ldots n} are all even, the rightmost position in a binary number (corresponding to \(2^0 = 1\)) determines whether the number is odd or even. So the problem reduces to: ‘is the last digit 1 (odd) or 0 (even)?’
a diagonal line, two crossing lines, and the lower end of an upright line. Whenever we can identify all those parts in a picture of a number, we can be fairly sure that the number should be four. With enough training examples, the CNN learns that this combination of features is strongly correlated with the number four. A zero, on the other hand, would have more curved features and no crossing lines. Some relevant features (again according to human intuition) are indicated in the image on the right in Figure 6.4. Presence of these features heightens the probability of the image being an example of the number zero, and lowers the probability of the image being an example of the number four. An attractive property of CNNs is that we do not need to specify any of these features. Rather, the network learns to find relevant patterns by itself. For more details on how this works for digit recognition, see LeCun et al. 1998.

Convolutional Neural Networks consist of multiple layers, where each layer learns more abstract patterns than the previous one (combining lower-level features). Convolutional layers recognize images by sliding filters over an image, and computing a function between the filter and the image at every step. This sliding is shown in Figure 6.4 by the image in the middle. After having computed the matches for all filters at all locations, we have a new grid of values that gets sent to the next layer. Following the convolutional layers, CNNs usually end with a fully connected layer (or a set of fully connected layers) that serves to make predictions about the input. With modern CNNs, the network architecture can become quite complex, as illustrated in Figure 6.5, which shows the CNN known as GoogLeNet (Szegedy et al., 2015). This network is over 30 layers deep, and uses multiple convolutional modules (with different filter sizes) per layer (all but five of the blue boxes in Figure 6.5).  

GoogLeNet won the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) in 2014 (Russakovsky et al., 2015). The goal of this challenge is for systems to correctly classify 1000 different types of objects, in a large collection of images. CNNs have become the standard approach to this task, since Krizhevsky et al.’s (2012) AlexNet system won the 2012 challenge with a 10% lower error-rate than the first runner-up (which used a set of hand-crafted features).

The success of AlexNet led other researchers to explore why CNNs are so successful, and what kind of features are learned by Convolutional Neural Networks. Zeiler and Fergus (2014) and Yosinski et al. (2015) visualize what different layers in image-labeling CNNs respond to. They show that the feature maps from the lower layers correspond to low-level features (corners and edges), while feature maps in higher layers correspond more closely to the different classes that the CNN is trained to recognize (dogs’ faces, birds’ legs). Soon after, researchers realized that CNNs trained for the ILSVRC could also be more broadly applied. For example, Donahue et al. (2014) trained a CNN on the 2012 ImageNet data, and then showed that the features

---

2The other blue boxes correspond to fully connected layers preceding the yellow object classification layers. There are two FC-layers for both of the ‘intermediate’ classification layers, and one FC-layer for the final classification.
Many neural network architectures, such as the Multilayer Perceptron, have a fixed input size. This makes it difficult to work with text data, because sentences can be arbitrarily long; there is no upper bound to how long a sentence can be. Recurrent Neural Networks (RNNs, Elman 1990) are designed to handle (text) sequences of arbitrary length. In recent years, RNNs have become one of the workhorses of Natural Language Processing. This section provides a general introduction to RNNs and how they are used. For a more extensive overview, see Lipton et al. 2015; Goodfellow et al. 2016; Goldberg 2017.

We will assume that text sequences are represented as lists of tokenized words (even though one might also choose to represent text as a sequence of characters). We can feed text into an RNN by providing the tokens one-by-one, in separate time steps. Alternatively, RNNs can also produce sequences of text, generating sentences word-by-word.

### 6.4.1 Model architecture

The basic RNN architecture is illustrated in Figure 6.6. It consists of an input $X$ (provided at time step $t$), an RNN unit, and an output $H$ (the Hypothesis at time step $t$). The RNN unit is connected to itself, which means that at every time step, it sends some information from its hidden state to itself as input for the next time step. Instead of representing RNN using this recursive loop, we can also present them unrolled as in Figure 6.7. This presentation shows the entire sequence of time steps.
6.4.2 Uses of RNNs

A basic use of RNNs is to apply them for sequence labeling tasks, such as Part-of-Speech tagging and Named Entity Recognition, where there is a one-to-one mapping between the input and the output. Table 6.1 shows an example sentence with its tokens associated with part-of-speech tags and entity labels. For every input token $X_t$, the RNN can use the preceding tokens $X_0, ..., X_{t-1}$ to decide upon the right tag or label for $X_t$. The predicted tag or label is the one that has the highest probability, given the current input and the sequential data observed so far.

<table>
<thead>
<tr>
<th>Tokens:</th>
<th>Keith</th>
<th>Richards</th>
<th>performed</th>
<th>in</th>
<th>Arnhem</th>
<th>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tags:</td>
<td>PROPN</td>
<td>PROPN</td>
<td>VERB</td>
<td>ADP</td>
<td>PROPN</td>
<td>PUNCT</td>
</tr>
<tr>
<td>Labels:</td>
<td>PERSON</td>
<td>PERSON</td>
<td>-</td>
<td>-</td>
<td>LOCATION</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 6.1 Table showing a mapping from the input (the tokenized sentence Keith Richards performed in Amsterdam) to possible outputs: either part-of-speech tags or entity labels. These kinds of mappings could be learned by an RNN model. (PROPN stands for proper noun, ADP for adposition, and PUNCT for punctuation.)

6.4.3 Different kinds of RNNs

Although basic RNNs work well for many sequence modeling problems, different researchers have proposed extensions or modifications to improve their performance.

Bidirectional RNNs. As Figure 6.7 shows, standard RNNs only operate in one direction: either from left to right, or from right to left. But whatever direction we go in, the RNN cannot use the next tokens ($X_{t+1}, ..., X_n$) to predict a label for the current token, even though that additional context could be very useful. Bidirectional RNNs (Schuster and Paliwal, 1997) solve this problem by having two RNNs operate over the input sequence: one that goes from left to right, another that goes from right to left. An additional layer uses the information extracted by both RNNs at the same time steps to make predictions about the input.

Gated RNNs. The problem with basic RNNs, as they were originally conceived, is that they struggle with longer dependencies; with long sequences, it is difficult for the network to ‘remember’ information from the beginning of the sequence, all the way up to the end (Bengio et al., 1994). This lead to the introduction of gated RNNs: recurrent neural networks where
the RNN modules have a memory component, with *gates* that determine whether to keep remembering, or to forget particular information. Gated RNNs learn by themselves (using the training data) how to control those gates. The most common gated RNNs are Long Short-Term Memory networks (LSTMs, Hochreiter and Schmidhuber 1997) and Gated Recurrent Units (GRUs, Cho et al. 2014).

### 6.4.4 Encoding and decoding sentences

RNNs are also used to produce representations of sequential data, that can be fed to other machine learning components, such as classifiers. An RNN-classifier can be used to predict properties of a sequence, e.g. whether a sequence of words forms a grammatical Dutch sentence, or whether the sequence carries positive or negative sentiment. Figure 6.8 provides an illustration. In this scenario, we can say that the RNN is used as an *encoder*.

![Figure 6.8](image)

**Figure 6.8** Recurrent Neural Network used to classify the polarity of a sentence as either positive or negative. (EOS) is a token that signals the end of the sentence. This example uses a multilayer perceptron (MLP, a neural network with at least three layers: an input layer, one or more hidden layers, and an output layer) to classify polarity, based on the output of the LSTM.

The reverse is also possible. RNN-decoders take vector representations as their input and produce sequences as their output. Those vector representations can also be generated by another RNN. This technique is often used for *sequence-to-sequence* (or *seq2seq*) problems such as Machine Translation. The idea (proposed by Cho et al. 2014 and Sutskever et al. 2014, illustrated in Figure 6.9) is that the message from one language is projected into a shared semantic space between the encoder and the decoder, and the decoder uses that representation to reproduce the message in another language.

Rather than decoding a message from one language into another, Vinyals et al. (2015) propose to use an LSTM-decoder to produce image descriptions based on vector representations of images. They use a pre-trained convolutional network model to compute feature vectors for the images in the Flickr30K and MS COCO image description datasets (Young et al., 2014; Lin et al., 2014), and train an LSTM to produce descriptions for those images, based on the extracted features. Figure 6.10 provides an illustration. Note that the image is only provided at the start of the generation process, rather than at every time step (as in Mao et al. 2015, for example).
Chapter 6  Automatic image description: a first impression

Figure 6.9 RNN used to translate a sentence. The RNN on the left is used to encode the Dutch source sentence, while the RNN on the right is used to decode the hidden representation into English. Note that the decoder uses the predicted words from each previous time step ($H_{t-1}$) to predict the next word. ⟨BOS⟩ is a token that signals the start of the sentence to be decoded.

Figure 6.10 RNN initialized with an image feature vector, and a beginning-of-sentence (BOS) token. The dashed lines indicate that the hypothesis $H_{t-1}$ may be used at inference time as the input for the next time step.

6.4.5 Attention mechanisms

Regular conditioned RNNs can only look at the image as a whole, because the visual feature vector does not contain any spatial information. The standard approach of using the penultimate layer of an image labeling CNN means that the feature vector only contains information about what is in the image, not where it is. This limits the kind of descriptions that these models can generate. After all: it is hard to talk about what you cannot see. To tackle this issue, Xu et al. (2015) present an image description system with an attention module, illustrated in Figure 6.11.4

The idea behind the attention module is that the system should learn to identify salient (visually important) parts of an image, and attend to those regions while describing them. To achieve this, the attention module receives two inputs: (1) a set of feature maps, corresponding

---

4Attention modules have also been used to improve machine translation systems (e.g. Bahdanau et al. 2015). The idea is that, for every word the system produces in the target language, it should have evidence from the source language. Attention-based machine translation models explicitly learn where to look for this evidence in the source sentence. For a more elaborate discussion, see (Olah and Carter, 2016).
Figure 6.11 RNN with an attention module. The model is first provided with a set of feature maps, corresponding to different regions of the image. At every time step, the attention module learns to identify relevant parts of the image for the system to describe. As its input, the module takes the feature maps and the RNN’s hidden state from the previous time step. The attention model produces a single feature map for the RNN to use in generating the next word.

Generative Adversarial Networks

Generative Adversarial Networks (GANs) were proposed by Goodfellow et al. (2014) for the problem of learning generative models of data. The idea is to train two neural networks that compete with each other: one tries to generate realistic images, while the other tries to discriminate between real and artificially generated images. This drives the generator to produce images that fall into the same distribution as the training images. GANs have been highly successful at generating realistic images and videos, and this success has led others to propose adversarial training for other applications as well. Recently, Dai et al. (2017) and Shetty et al. (2017) have proposed different GAN-based image description systems. What makes GANs successful at producing more diverse descriptions is the presence of an additional objective: not only do they have to be good at predicting the next word at every time step, but they also have to make sure that the entire description is human-like as well.
6.6 Takeaway

The previous sections (§6.2-6.5) discussed the building blocks for data-driven image description systems (based on neural networks) that have become standard for automatic image description. Understanding how current systems work also helps to see (at an abstract level) how they could be improved in future research:

1. Improve the visual component. If we can extract better features from the image, or information about the image, then the descriptions may become more reliable and accurate. One way to approach this problem is to design models that perform better on the ImageNet Visual Recognition Challenge, and then use their internal representation of the image (rather than representations from existing feature extractors). See Kornblith et al. (2018) for a discussion of this idea.

2. Improve the generation component. If we change how to act on the visual information, we may be able to produce higher quality descriptions. Possible changes are:
   - Change the kind of feedback the system receives while training. GAN-based models are an example of this. These models don’t necessarily perform better than other models (as measured by BLEU, Meteor, and other automated metrics), but they do generate more diverse descriptions (Dai et al., 2017; Shetty et al., 2017, see also Chapter 7 of this thesis).
   - Add other sources of information for the system to use in the description process. This idea has been explored in the context of Visual Question Answering (Antol et al., 2015), see e.g. (Wu et al., 2017a).

At the same time, the limitations of current data-driven image description systems are also clear: they don’t do much more than correlate image features with sequences of words. Judea Pearl, in an interview (Hartnett, 2018) about his recent book (Pearl and Mackenzie, 2018) calls this *curve fitting*. He argues that, for real intelligent behavior, we need systems to reason about the world. This has traditionally been the domain of more formal, rule-based systems (which tend to be restricted to a small domain, because rule-writing is very labor-intensive). It is unclear where the field is going, but these are fertile grounds for the development of hybrid systems that enjoy the best of both worlds.

6.7 Evaluation

We only know how good or bad a system is once we have evaluated it. The question of how to evaluate Natural Language Processing systems has a surprisingly short history; just thirty years ago, system evaluation was considered a controversial topic (Paroubek et al., 2007). Nowadays, no NLP engineering paper is published without some form of evaluation, and automatic image description is no exception.

6.7.1 Evaluation of automatic image descriptions

As Bernardi et al. (2016) note in their survey of the image description literature, automatic image description systems have been evaluated in two ways: either through human judgments or through automated metrics. We briefly discuss each of these below.
Human judgments

Early work on image description was evaluated with text-based similarity measures and a human judgment study (Bernardi et al., 2016). This type of judgment study involves asking humans to rate whether the descriptions accurately describe the image, are grammatically correct, are relevant for the image, are human-like, inter-alia, using a Likert-scale survey. The main criticisms of human judgment studies is they are expensive to perform and difficult to replicate without access to the same subject pool and control samples (e.g. Papineni et al. 2002; Hodosh and Hockenmaier 2016). Nevertheless, these studies are the clearest indication of overall performance differences between models.

Automatic evaluation

Recent advances in automatic image description have mostly been evaluated with text-based similarity metrics. These metrics compare automatically generated descriptions (the hypotheses) for a set of images with the (human-generated) reference descriptions associated with those images. Jurafsky and Martin (2009) note that the intuition behind these metrics “derives from Miller and Beebe-Center (1958), who pointed out that a good MT output is one that is very similar to a human translation.” Examples are:

BLEU (Papineni et al., 2002) computes the amount of n-gram overlap between the hypothesis and the reference descriptions, using a modified n-gram precision metric. In other words, BLEU asks: to what extent can we find the same n-grams from the hypothesis in the reference descriptions?

ROUGE (Lin, 2004) computes the extent to which the hypothesis overlaps with the references, using a recall-based approach. In other words, ROUGE asks: how much of the information in the references is also captured by the hypothesis?

TER (Snover et al., 2006) computes the minimum amount of edits needed to transform the hypothesis into the closest reference.

Meteor (Banerjee and Lavie, 2005; Denkowski and Lavie, 2014) is similar to BLEU and ROUGE but adds the ability to match synonyms and paraphrases, using WordNet and a paraphrase table.

Metrics like these make it easy for researchers to benchmark the effect of their modeling decisions in terms of overall quality, but they are not informative about the strengths and weaknesses of a proposed model. This is especially true for n-gram based metrics, such as BLEU, which measure grammatical fluency and not semantic adequacy (Reiter and Belz, 2009).\(^5\) Elliott and Keller (2014) show that BLEU, Meteor, ROUGE and TER have at best a moderate correlation with human ratings of image description quality. More recently, different researchers have proposed other metrics for image description evaluation:

CIDEr (Vedantam et al., 2015) is similar to existing metrics that compare a hypothesis with a set of reference descriptions, except that it gives a higher weight to words that are more informative (as computed using the TF-IDF score for each word).

\(^5\)We may also note the current trend to highlight the shortcomings of BLEU in particular, e.g. with Reiter’s (2018) structured review of the validity of BLEU, and Sulem et al.’s (2018) analysis showing that BLEU is not suitable for text simplification evaluation.
SPICE (Anderson et al., 2016) uses the reference descriptions to build a ‘scene graph’, which can be represented as a set of propositions about the picture. Automatically generated descriptions are also parsed into a scene graph, and the SPICE metric measures the extent to which the two scene graphs overlap (expressed as an F1-score, combining precision and recall over the propositions).

Word Mover’s Distance (WMD; Kusner et al. 2015) was originally developed to measure document similarity. Kilickaya et al. (2017) modified this metric for the evaluation of image descriptions. Rather than directly working with the tokens in the hypothesis and references, the WMD metric uses word embeddings to compute the distance between the hypothesis and each individual reference description.

Kilickaya et al. (2017) show similar results to Elliott and Keller’s (2014) study. For all metrics listed above (except for TER), they found that these metrics have at best a moderate Spearman correlation (between 0.44 and 0.64) with human judgments. Furthermore, the authors find that the different metrics seem to capture different aspects of description quality. In particular, they note that Meteoor, SPICE, and WMD seem to complement each other; after combining these metrics, the authors obtain a Spearman correlation of 0.66 with human judgments. Having that said, there is still room for improvement of these metrics. We will discuss some possibilities in Section 6.13.1.

6.8 Error analysis

Error analysis is the process of identifying the mistakes that a system makes, and ordering those mistakes into coherent subgroups. This categorization reveals the distribution of the different kinds of errors, so that we know (if we used a representative sample) which errors occur most often, and which occur less frequently. The remainder of this chapter presents a coarse- and fine-grained analysis of the descriptions generated by a state-of-the-art attention-based model (Xu et al., 2015), trained on the Flickr30K dataset (Young et al., 2014). The goal is to assess the qualities of a state-of-the-art model to illustrate the recent progress in this area and the challenges that lie ahead.

6.8.1 Coarse-grained analysis

Our coarse-grained analysis quantifies whether the descriptions are accurate or inaccurate. (We define accurate to mean that the description is congruent with the image, without it necessarily being the “best” or most complete description.) This is similar to the human judgment studies discussed earlier. Our coarse-grained analysis is a binarized version of the correctness scale from Mitchell et al. (2012). Figure 6.12 provides some examples of image descriptions with differing amounts of errors. Each of these would be classified as INACCURATE.

6.8.2 Fine-grained analysis

Our fine-grained analysis takes the image descriptions that have been classed as INACCURATE, and further classifies them in terms of our taxonomy of errors, presented in Section 6.9. This gives us an indication of the distribution of errors that the system produces.

Their Table 3 shows correlations with human judgments on the Flickr8k dataset from Elliott and Keller 2014. The same table also shows that all metrics have weaker Spearman correlations with data from Aditya et al. (2015) (between 0.39 and 0.44), but these numbers conflate correctness and thoroughness.
One error

A woman in a **red** shirt is standing in front of a building

Two errors

A man in a **yellow** helmet rides a **bike** in the air

Three errors

A blond **woman** in a **white** shirt is **blowing** her teeth

Four errors

A little **boy** in a **white shirt** playing soccer

**Figure 6.12** Examples of images with 1–4 errors. The annotated errors are marked in boldface. Original images by: Feggy Art (CC BY-NC-ND 2.0), el Reino (All rights reserved), Edbury Enegren (CC BY-NC-SA), and Neil Smith (CC BY-NC-SA), all through Flickr.com.

Our work is most closely related to earlier work by Hodosh and Hockenmaier (2016), who propose an evaluation of image description systems using binary forced-choice tasks, where systems have to choose the best description for a given image. For each image, the system can choose between the original description or a manipulated description. By controlling the manipulations, the authors are able to check for weaknesses in image description systems. Their error categories (i.e. the different kinds of manipulations) partially overlap with ours, though we provide a more fine-grained typology.

**6.9 Error categories**

We developed a non-exhaustive categorisation of errors by inspecting the descriptions generated by an attention-based image description model (Xu et al., 2015). We trained the model on the Flickr30K dataset (Young et al., 2014), with 300-dimensional word embeddings, a 1000-dimensional GRU hidden layer (Cho et al., 2014), and ‘CONV\(_{5,4}\)’ image features from the VGG-19 convolutional neural network (Simonyan and Zisserman, 2015). We generated 1,014 descriptions with a beam width of five hypotheses, recording a Meteor score of 17.4 on the
Flickr30K test set. All our code and data is available online.\footnote{See: https://github.com/evanmiltenburg/ErrorAnalysis}

In total, we identified 20 common types of errors, which we grouped into four main categories: People, Subject, Object, and General. We developed annotation guidelines with examples for each type of error. The error categories and types of errors are described below. For the full guidelines, see Appendix D.

**People** Image description models often make mistakes that are specific to the description of people. Types of errors in this category are age (e.g. woman instead of girl), gender (man instead of woman), type of clothing (shirt instead of jacket), and color of clothing (red shirt instead of blue shirt).

**Subject** Mistakes relating to the subject of the description. This category contains the following types of errors: wrong when the wrong entity in the image is chosen as the subject, similar when the model mis-identifies the subject for something visually similar (e.g. guitar instead of violin), non-existent when nothing close to the mentioned entity is present in the image, and extra subject when an additional (nonexistent) entity is described along with the correct entity.

**Object** Similar to Subject.

**General** Mistakes that are not specific to people. Error types in this category are: stance for posture-related mistakes, activity for wrongly identified activities, position for mistakes in spatial relations within the image, number for counting errors (too few/many entities mentioned), scene/event/location for mis-identifications of the scene, event, or location, color for non-clothing entities that are mistakenly attributed with a color, other for any unforeseen mistakes, and generally unrelated for descriptions that do not seem to have any relation with the image. In these cases, it is impossible for annotators to assign any error category to the description. E.g. if the first image in Figure 6.12 were to be described as *A dog runs through the snow.*

### 6.10 Annotation tasks

We define two error annotation tasks: The coarse-grained annotation task is a binary categorization problem, where an annotator determines for every description whether it is accurate. The fine-grained annotation task is a multiclass categorization problem, given the error types presented in the previous section. Each inaccurate description is annotated with one or more error types. We can think of this task as a means to assess the semantic edit distance between a generated description and the closest accurate alternative.

In total, one annotator categorized all 1,014 generated descriptions into the coarse-grained groups: accurate and inaccurate descriptions. The same annotator then performed the fine-grained annotation. We validated the annotation scheme by double-annotating a random selection of 100 descriptions (10% of the data, annotating both coarse and fine-grained) to determine whether the annotation guidelines provide a reliable basis for annotating the errors.

#### 6.10.1 Results for the coarse-grained task

In the coarse-grained annotation task, 812 out of 1014 descriptions (80%) were judged to be inaccurate. We achieved a good inter-annotator agreement of Cohen’s $\kappa=0.67$, with an accuracy of 91%. The discrepancy between these numbers is explained by the label distribution: the inaccurate category is so dominant that any disagreement yields a high penalty in $\kappa$. Out
of the 100 double-annotated descriptions, the first and second annotator judged 86 and 81 descriptions to be inaccurate, with agreement on 79 descriptions.

### 6.10.2 Evaluating the fine-grained annotations

We found 1,265 errors in 812 descriptions, which is an average of 1.56 errors / description. Tables 6.2 and 6.3 show the number of errors per image, and the distribution of error types across the dataset. Surprisingly, the most common error category is generally unrelated (264 times). Errors from the general and people categories are much more frequent than the other two. Taken together, the subject category is least common. Our intuition is that this is because mistakes in decoding the subject from the language model affect the entire sentence; the choice of subject influences the probability of all subsequent words, leading to a generally unrelated sentence.

<table>
<thead>
<tr>
<th>Number of errors</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>486</td>
<td>221</td>
<td>83</td>
<td>22</td>
</tr>
</tbody>
</table>

**Table 6.2** The distribution of error annotations. Top: the number of errors for a single description. Bottom: how many descriptions have exactly that many errors.

<table>
<thead>
<tr>
<th>Type</th>
<th>Count</th>
<th>Type</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>generally unrelated</td>
<td>264</td>
<td>non-existent object</td>
<td>47</td>
</tr>
<tr>
<td>color of clothing</td>
<td>195</td>
<td>age</td>
<td>40</td>
</tr>
<tr>
<td>activity</td>
<td>168</td>
<td>stance</td>
<td>38</td>
</tr>
<tr>
<td>type of clothing</td>
<td>104</td>
<td>position</td>
<td>37</td>
</tr>
<tr>
<td>gender</td>
<td>98</td>
<td>extra subject</td>
<td>34</td>
</tr>
<tr>
<td>scene/event/location</td>
<td>91</td>
<td>similar-object</td>
<td>31</td>
</tr>
<tr>
<td>number</td>
<td>61</td>
<td>other</td>
<td>20</td>
</tr>
<tr>
<td>non-existent subject</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wrong-object</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>similar-subject</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>extra object</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wrong-subject</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>color</td>
<td></td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>non-existent subject</td>
<td></td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>wrong-object</td>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>similar-object</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>extra object</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>wrong-subject</td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

**Table 6.3** Number of times each error was annotated in our fine-grained analysis.

The fine-grained annotation task is inherently ambiguous because inaccurate descriptions might be corrected in many different ways. The first image in Figure 6.12 illustrates this ambiguity. The generated description for this image is given in Example (33a). This description could either be corrected to (33b) or (33c), depending on whether one assumes the mistake is in the color or the type of clothing.

(33) a. A woman in a **red shirt** is standing in front of a building
    b. A woman in a **black shirt** is standing 
    c. A woman in a **red skirt** is standing ...

Subjectivity and ambiguity are inherent to the task of image description; describing an image in one simple sentence means that you have to make a choice about what to include in your description. But this subjectivity also means that it is difficult to provide a proper intrinsic evaluation for the annotation task: different choices about how to describe an image may be equally valid. To quantify the extent of this issue, we treat the double annotation for the fine-grained task as a retrieval problem, i.e. how many error types are also found by the
Table 6.4 Error categories and the BLEU-4 and Meteor scores after correcting the errors. $\Delta$ indicates improvement in the scores between the modified descriptions and the original descriptions.

<table>
<thead>
<tr>
<th>Type</th>
<th>BLEU</th>
<th>$\Delta$</th>
<th>Meteor</th>
<th>$\Delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>17.8</td>
<td>—</td>
<td>17.2</td>
<td>—</td>
</tr>
<tr>
<td>Color of clothing</td>
<td>18.8</td>
<td>1.0</td>
<td>17.5</td>
<td>0.3</td>
</tr>
<tr>
<td>Activity</td>
<td>18.5</td>
<td>0.7</td>
<td>17.7</td>
<td>0.5</td>
</tr>
<tr>
<td>Type of clothing</td>
<td>18.1</td>
<td>0.3</td>
<td>17.4</td>
<td>0.2</td>
</tr>
<tr>
<td>Gender</td>
<td>18.6</td>
<td>0.8</td>
<td>17.6</td>
<td>0.4</td>
</tr>
<tr>
<td>Scene/event/location</td>
<td>18.0</td>
<td>0.2</td>
<td>17.4</td>
<td>0.2</td>
</tr>
</tbody>
</table>

second annotator? For the fine-grained annotation task, we ended up double-annotating 79 descriptions that both annotators agreed contained at least one inaccuracy. For these cases, we achieved a precision of 0.54, with a recall of 0.55. Based on this observation, we decided to carry out an extrinsic evaluation: how useful are the fine-grained annotations for guiding future research on model development? We discuss this evaluation below.

6.11 Correcting the errors

Now we have observed the frequency of each type of error, we can ask: what is the effect of addressing these errors on the automatic evaluation metrics? We selected the five most common error types (excluding generally unrelated), and manually corrected each error without looking at the reference descriptions. If a description is annotated with multiple errors, we only correct the relevant error. We tried to be conservative in our corrections; e.g. for color of clothing errors, if the system wrote e.g. \textit{white shirt} instead of \textit{checkered/leopard print/shirt}, we left the description untouched, rather than insert the pattern. For the activity errors, we tried to change as little as possible but editing the activity often also entails changing the object as well. For example, a sentence that read \textit{A man in a suit is holding a sign.} was changed to \textit{A man in a suit is talking.} because the man wasn’t holding anything and leaving out the object would produce an ungrammatical sentence. If a change would entail completely re-ordering the sentence, we leave the generated description untouched.

Table 6.4 presents the BLEU and Meteor scores for the validation set before and after correction. For example, after only correcting the colors of clothing, we find a one-point improvement for the BLEU score with respect to the original model.

We did not investigate whether these effects are cumulative, i.e. what happens if we correct all errors. Presumably, they are cumulative, but this task is not suitable for such an investigation because the corrections need to be restrictions in order for the improvement estimation to be accurate. If we allowed annotators to correct all the errors in a sentence, we would be giving them carte blanche to rewrite everything, turning the analysis into an evaluation of human performance.

6.12 Takeaway

Sections 6.8–6.11 provided an extensive error analysis for image descriptions generated by a state-of-the-art attention-based model. The main contributions of this analysis are:

1. Providing a taxonomy of common errors in automatically generated image descriptions.
2. Quantifying the weaknesses of the model. We posit that any model with a similar architecture will have similar weaknesses.

3. Quantifying the possible improvement of this model if those weaknesses are addressed.

We focused on the nature of the inaccurate descriptions, and looked at different errors that these contain. But what about the accurate descriptions? The descriptions that are accurate, are also much more general than the human descriptions, which usually include small, but salient details. We propose the following rule: if the majority of the human descriptions comments on an aspect of the image that is not addressed by a generated description, then that aspect could be improved. This idea is operationalized in the next chapter, where we propose the local recall metric (§7.4.2).

We see two other perspectives to build on the observations from this error analysis.

Automated error analysis: As noted earlier, Hodosh and Hockenmaier (2016) carried out a study in which they evaluate image description models using binary forced-choice tasks, where models have to choose which description best describes a particular image. The choices are carefully manipulated, so that each task evaluates the model’s performance in one area (e.g. recognizing scenes). Our taxonomy of errors could be used to extend the range of available tasks, for example with a task to evaluate the use of color terms;

Extending existing models: Table 6.4 provides an indication of how much a model could improve by incorporating a dedicated module to detect color, actions, type of clothing, gender, and scenes. We expect that our work will encourage researchers in vision & language to investigate this possibility. More generally, we hope that our taxonomy of error types will help others to go beyond similarity-based metrics, and to look at their model’s output through a qualitative lens.

Our results cast doubt on some of the findings from Anderson et al. (2016). Their paper, proposing the SPICE metric, argues that we can use SPICE to evaluate model performance in more detail than ‘global’ metrics like BLEU and Meteor. Specifically, the authors claim that SPICE is useful to evaluate whether models are able to identify relevant objects, relations between objects, and whether objects have particular attributes. For attributes, the authors identify three subcategories: color, count, and size. Referring to their Table 2 (comparing different system outputs with human performance), Anderson et al. (2016) argue that the models from Fang et al. (2015) and Vinyals et al. (2015) “outperform the human baseline in their use of object color attributes.” This is a surprising result, given our findings with Xu et al.’s (2015) attention-based model (which has been shown to outperform both Vinyals et al.’s (2015) and Fang et al.’s (2015) model). Humans are not likely to make the same mistakes as in Figure 6.12 (e.g. saying white shirt instead of black shirt), and we found many errors like this. Furthermore, we have to ask ourselves what it means to ‘outperform the human baseline’ on the SPICE metric. Participants of the image description task and image description systems are asked to do two different things. Humans receive very few examples of ‘proper’ descriptions, and produce texts about the images that capture the main contents of those images, based on their individual ideas of what a description should look like. Systems receive a large amount of training data, and are asked to produce descriptions that are similar to what they have seen before. Thus, their task is to generate ‘average’ descriptions that are close to what human participants have produced before. Because automated metrics evaluate descriptions based on human reference data, they are biased towards the image description systems, whose task is closer to the how they are evaluated. We conclude that ‘outperforming the human baseline’ in terms of the SPICE metric may not be a good indicator of actual performance, and that
human-level performance has not been achieved yet.\footnote{However, we should acknowledge the difference between \textit{models} and \textit{architectures}. Xu et al. (2015) have shown that they were able to train a well-performing model using their attention-based architecture. Although the model that we analyzed has the same architecture as Xu et al.'s (2015) model, it is a different model. And although it is plausible that Vinyals et al.'s (2015) and Fang et al.'s (2015) models would make similar mistakes as our model, it is not yet certain that they do. If we want to conclusively show that both Vinyals et al.'s (2015) and Fang et al.'s (2015) model still do not perform at human level, with regard to object color attributes, we should carry out another error analysis with the model outputs as they were evaluated by Anderson et al. (2016).}

6.13 Conclusion

This chapter presented an introduction to automatic image description, focusing on data-driven models. The first part of this chapter showed the building blocks that form the core of these models, while the second part of this chapter showed the shortcomings of one particular instantiation. It is clear that current approaches to automatic image description still leave plenty of room for improvement. If many of the generated descriptions seem generally unrelated to the images, and a recent model makes a substantial amount of errors in identifying color of clothing, then we are still a long way from the kind of reasoning about the images that we see in human descriptions (as shown in the first part of this dissertation).

6.13.1 Implications for image description research

Even though image description systems try to find the descriptions with the highest probability, given the input image, the error analysis shows that the generated descriptions are not necessarily faithful to the images themselves. This raises the question: how could we force image description models to remain faithful? (Aside from using better image representations, to reduce the noise in the input.)

We have already seen one approach in Chapter 2, when we were discussing ways to tackle bias in image description (§2.11). Burns et al.'s (2018) Equalizer model forces itself to use correctly gendered terms, and if the model finds no evidence of gender in the image, it uses a gender-neutral term (e.g. person, snowboarder). Another proposal was recently made in the \textit{Shortcomings in Vision & Language} workshop, where researchers in Vision & Language discussed weaknesses of current systems combining Computer Vision and Natural Language Processing. Madhyastha et al. (2018) note that current image description evaluation metrics do not take the images into account. Rather, they compute the similarity between the generated description and a set of reference descriptions. Optimizing for these metrics will not improve the accuracy of the generated descriptions. Instead, Madhyastha et al. suggest to use metrics that take image content into account as well. They propose to use pre-trained object detectors, and to compare generated image descriptions with the set of detected objects. While this is still a crude metric (for example, it does not take actions into account), it does show us a way forward to make descriptions more closely match the images they are supposed to be describing.

6.13.2 Next chapter

Having looked at the \textit{accuracy} of automatically generated image descriptions, the next chapter will also look at the \textit{diversity} of automatic image descriptions. We will compare the output of
nine different systems with human reference data. As we will see, automatic image descriptions use a much smaller vocabulary, and are much less diverse than their human-generated counterparts.