Stanford Machine Learning - Week XI

Eric N Johnson

September 17, 2016

1 Photo OCR

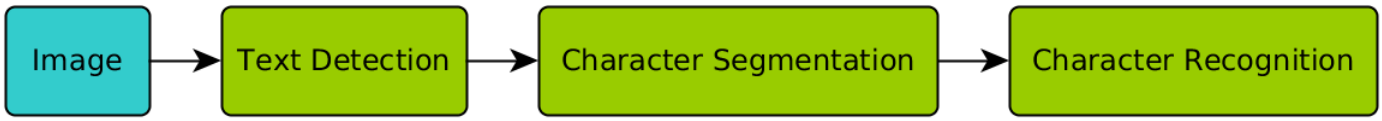
Photo OCR: Photo Optical Character Recognition
We use a computer to read and understand text in digital photographs.

1. Text detection
2. Character segmentation
3. Character classification

Some OCR systems do spelling correction at the end.

1.1 Problem Description and Pipeline

Here is a photo OCR pipeline:

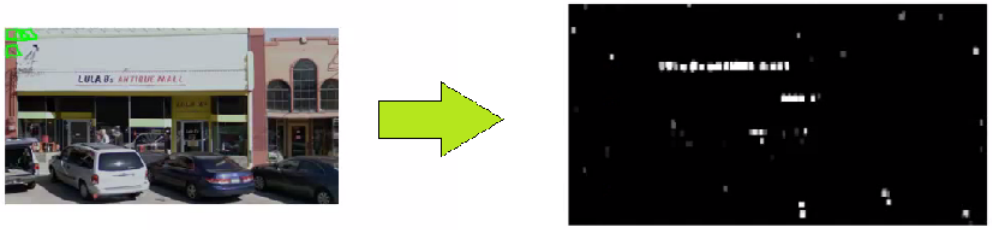


Each of these {Text Detection, Character Segmentation, Character Recognition} is handled by a separate module. The modules may be individual machine learning tasks. Deciding what pipeline will be used is one of the most important design decisions when creating a machine learning system. Several engineers may be tasked with implementing each part.

1.2 Sliding Windows

A *sliding windows classifier* is a rectangular region of fixed width and height aspect ratio that "slides across an image. The ML algorithm uses these windows to classify what is in the photo. The *step size* parameter or *stride* is used to slide the window across the entire image. Typically this is done with a small rectangle first, then windows of increasing size. Here is how Photo OCR works:

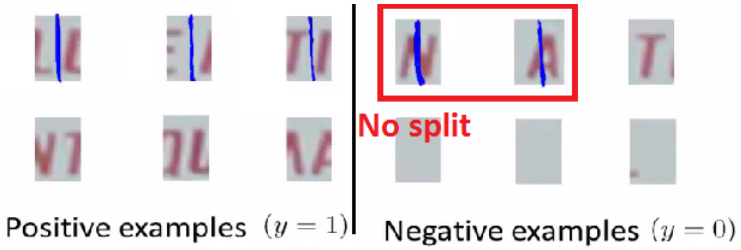
1. Text Detection:
   (a) A labeled training set is created with positive examples (with text, $y = 1$), and negative examples (without text, $y = 0$).
   (b) A sliding windows classifer is used to detect text in the image.



(c) An expansion operator is used to expand the areas where text was detected.

2. Character Segmentation:

   (a) Train a classifier to recognize when a set of letters is split within a window.



   (b) The splits are then thrown out of the set so that only whole letters are retained.

3. Character Recognition:

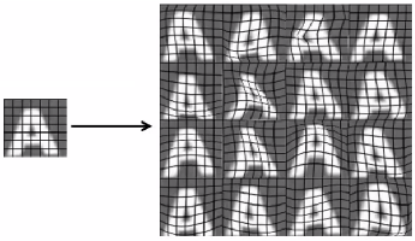
   (a) Use a neural network or other multiclass classification algorithm to recognize individual letters.

1.3 Getting Lots of Data and Artificial Data

One of the best ways to get a machine learning algorithm working well is to train a low-bias algorithm on many training examples. But where do you get the data to train your algorithm?

- We can enlarge existing training sets artificially...
- Or we can synthetically produce training sets from scratch

For example, in character recognition, we can produce many examples of letters using different fonts. Paste characters over different backgrounds, add blur, distort, and skew characters. Another way is to take a single example then distort, manipulate, and skew it. That single example can quickly become several.



In general it is not helpful to add random "meaningless" noise - but it is useful to consider what kind of distortions you may actually see in real data then try to replicate these.

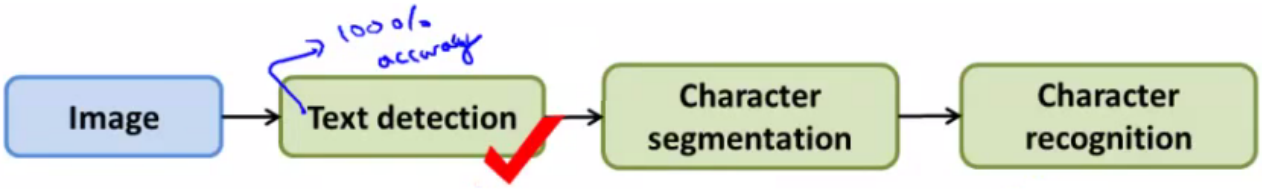
Getting more data:

1. Make sure you have a low bias classifer before expending the effort to synthesize more training examples. Low bias is crucial here.
2. Ask "how much work will it take to get 10x as much data as we currently have?" If we can get this data we probably should.
3. Collect and label data yourself. Figure out how much time it would take to do this - it may be worth doing.
4. Can you crowd source training data? Can you employ labelers to do this inexpensively. See "Amazon Mechanical Turk."

1.4 Ceiling Analysis: What part of the pipeline to work on next

The most valuable (and expensive thing) is the analysts time! We can use *Ceiling Analysis* to determine what the best use of time is. What part of the pipeline should you spend the most time trying to improve? Single real number metrics on learning algorithm success are most useful.

- Simulate 100% accuracy of each part. For example, using our Photo OCR algorithm we can simulate 100% accuracy on the text detection part by feeding it only positive examples.



By doing this we can see:

- Will doing this increase in the accuracy of other parts of the pipeline?
- Determine when spending a lot of time on part of the pipeline *will not* advance the overall system accuracy.
- Using this idea we can find upper bounds for how good our algorithm could be - and what the gains may be if we spend more time on one component of the pipeline.