2017-04-20

Tom Rochette <tom.rochette@coreteks.org>

July 24, 2025 — daae079c

0.1 Context

0.2 Learned in this study

0.3 Things to explore

- How are embeddings computed?

1 Problems faced

2 Overview

- Embeddings are a form of dimensionality reduction, you specify the number of components/features you want as output
- For instance, given a word, the embedding function E will convert a string S of n characters into a vector of m dimensions
  - One could have created a mapping of words to a given integer, such that $\mathbb{N}^n \to \mathbb{N}$, where $\mathbb{N}^n$ is the sequence of bytes that represent a word (e.g., Hello → [H, e, l, l, o] → [104, 101, 108, 108, 111] → 226) and $\mathbb{N}$ is a unique identified of that number sequence (we could also have concatenated the bytes of the word Hello to make the number 104101108108111, but given that a language has (generally) a limited set of words, we can make more efficient use of space by using an incremental list of number)
  - The main issue with such approach is that the image value is simply a value that was randomly assigned to the word and thus no relation can be described using the number alone (other than saying "word 225 was created before 226, and word 227 was created after word 226")
  - But if you start to increase the number of dimensions, you begin to create dimensions in which a degree of ordering can begin to exist
  - It would also be possible to use a mapping such as $\mathbb{N}^n \to \mathbb{R}$, which has the benefit of allowing you to add new words in-between existing words, which is not something you could do with natural numbers. You would have to displace all existing numbers to insert your new word in-between two words, and thus, reassign numbers to words that already had a number assigned to them
    - One issue with mapping to reals is that as you want to insert a value in-between two values, you will require more and more precision. Given that you decide to use the range [0, 1], as you add words, the initial assignments will be 0, then 1, and then depending on how "related" the next word is to the word represented by 0 or 1, it will be closer to one or the other (or end up being 0.5)
- An embedding is sometimes annotated as $X \hookrightarrow Y$
- An embedding should generally try to preserve linearity (two points in the initial set should have the same distance in the embedded set)
  - But the process of embedding may lead to loss of information, such as trying to embed a circle onto a line

2.1 Handwriting recognition

I worked on converting the IAM XML form into the Pascal VOC format so that I could use YOLO: Real-Time Object Detection, but more specifically the TensorFlow implementation, darkflow. After converting a few IAM XML form files, I wanted to start training the model, but it seems that the yolo and yolo-tiny configurations are too big even for my GPU with 2 GB of RAM and training on the CPU is extremely slow. Below is the network detail of yolo-tiny, with the last maxpool and last 2 convolutional layers (1024 filters) removed.

Source	Train?	Layer description	Output size
		input	(?, 416, 416, 3)
Init	Yep!	conv 3x3p1_1 +bnorm leaky	(?, 416, 416, 16)
Load	Yep!	maxp 2x2p0_2	(?, 208, 208, 16)
Init	Yep!	conv 3x3p1_1 +bnorm leaky	(?, 208, 208, 32)
Load	Yep!	maxp 2x2p0_2	(?, 104, 104, 32)
Init	Yep!	conv 3x3p1_1 +bnorm leaky	(?, 104, 104, 64)
Load	Yep!	maxp 2x2p0_2	(?, 52, 52, 64)
Init	Yep!	conv 3x3p1_1 +bnorm leaky	(?, 52, 52, 128)
Load	Yep!	maxp 2x2p0_2	(?, 26, 26, 128)
Init	Yep!	conv 3x3p1_1 +bnorm leaky	(?, 26, 26, 256)
Load	Yep!	maxp 2x2p0_2	(?, 13, 13, 256)
Init	Yep!	conv 3x3p1_1 +bnorm leaky	(?, 13, 13, 512)
Init	Yep!	conv 1x1p0_1 linear	(?, 13, 13, 30)

In the process, I had to fix the code that collected the Pascal VOC XML files since it wasn't working properly (thus preventing training).

My idea was to use the IAM data set to indicate characters and have darkflow first start by showing me it can recognize characters in an image or video. Then, my next step would be to have it become able to recognize the specific characters.

2.1.1 Notes

https://github.com/thtrieu/darkflow/issues/148

If you get the following error `Input to reshape is a tensor with X values, but the requested shape requires a multiple of Y`.

The number of filters of your last layer has to be `filters = #num * (#classes + 5)`

2.1.2 Training darkflow

To train darkflow, you have to do the following

1. Edit `labels.txt` at the root of the git repository
2. Create and configure your custom network architecture `cfg/network.cfg` (you can copy `yolo.cfg` or `yolo-tiny.cfg`)
3. Prepare a set of annotations files (which will be in the annotations folder) in the Pascal VOC XML format, which has this minimal format

Where

- `filename` is the name of the input image file in your dataset folder
- `size.width` is the width of the input image
- `size.height` is the height of the input image
- `object.name` is the name of an object in the image and the class you want to match with in the `labels.txt` file (e.g., dog, cat, person, train, car)

- `object.xmin` is the minimum x coordinate of an object in the image
- `object.ymin` is the minimum y coordinate of an object in the image
- `object.xmax` is the maximum x coordinate of an object in the image
- `object.ymax` is the maximum y coordinate of an object in the image

An image may have multiple `object` entries.

4. Call `python flow --train --config cfg/network.cfg --dataset path/to/dataset --annotation path/to/annotations`. You may add `--gpu 1.0` if your GPU has enough RAM to run the network on it.

3 See also

4 References

- https://en.wikipedia.org/wiki/Embedding
- https://en.wikipedia.org/wiki/Word_embedding
- http://sebastianruder.com/word-embeddings-1/
- https://github.com/thtrieu/darkflow