Text Generation with LSTM Units

By Me

November 9, 2017

Goals

- Have fun.
- Replicate the results shown in the blog post “The Unreasonable Effectiveness of Recurrent Neural Networks”.

How Neural Networks Work

- In general:

$$f_w : [-1,1]^n \longmapsto [-1,1]^m$$

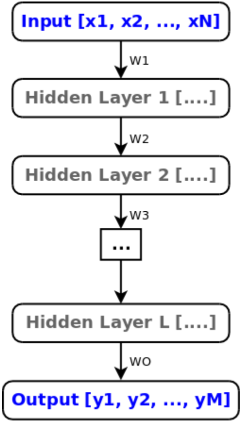
How Neural Networks Work

- A neural network consists of layers.
- The first layer receives the input, transforms it, and passes it on to the next layer.
- Each subsequent layer receives an array from the previous layer, transforms it, and passes it on.
- The output of the last layer is the output of the neural network.

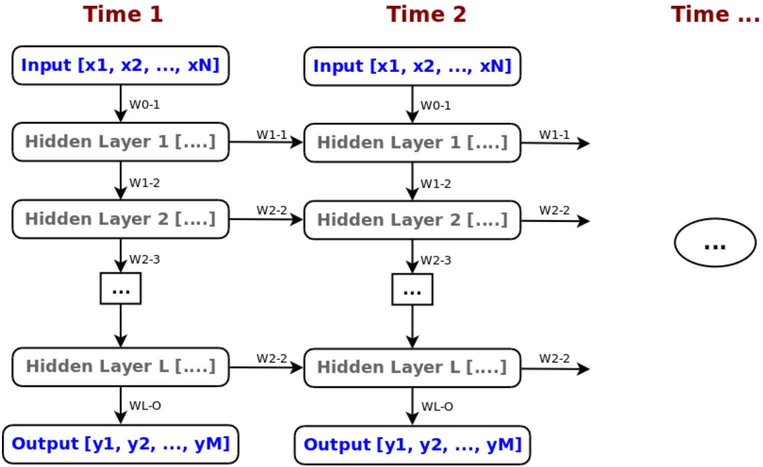
How Neural Networks Work

- A neural network consists of layers.
- The first layer receives the input, transforms it, and passes it on to the next layer.
- Each subsequent layer receives an array from the previous layer, transforms it, and passes it on.
- The output of the last layer is the output of the neural network.
- Transformation: Multiply the input with a matrix and apply a function to each element of the result.

How Neural Networks Work



How Recurrent Neural Networks Work



The Input

- The neural network is fed characters one at a time.
- For this, a mapping from characters to an array of size N is needed.
- I used the *one hot* method.

One Hot

- If the input is one out of N characters, an array of size N is needed.
- Characters are numbered.
- Each element is represented by an array of one one an N-1 zeros, where the one is at the position of the characters number.

One Hot: Example (alphabet)

- $a \longmapsto [1, 0, 0, 0, \ldots, 0, 0]$
- $b \longmapsto [0, 1, 0, 0, \ldots, 0, 0]$
- $c \longmapsto [0, 0, 1, 0, \ldots, 0, 0]$
- $\ldots$
- $y \longmapsto [0, 0, 0, 0, \ldots, 1, 0]$
- $z \longmapsto [0, 0, 0, 0, \ldots, 0, 1]$

The output

- The neural network is trained to predict the next character.
- It is given the *one hot* representation of the character for training.
- After training the output is a probability distribution over the chatacters.

What Kinds of Layers are Needed

- I needed three kinds of layers:
  - Tanh Layers.
  - LSTM Layers.
  - Softmax Layers.

How Does a Tanh Layer Work

- It is a simple mapping: $f_w : [-1,1]^n \longmapsto [-1,1]^m$.

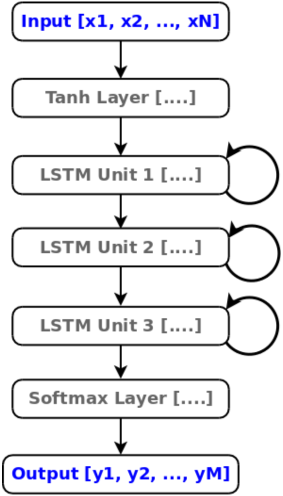
How Does a LSTM Layer Work

- It is also a mapping: $f_{w,s} : [-1, 1]^n \longmapsto [-1, 1]^m$.
- But it has an internal state, meaning that previous runs of the neural network may influence the output.

How Does a Softmax Layer Work

- It is a mapping: $f_w : [-1,1]^n \longmapsto [0,1]^m$.
- Where the output values add up to one, so that the output can be interpreted as a probability distribution.

How Do I Combine them to Generate Text



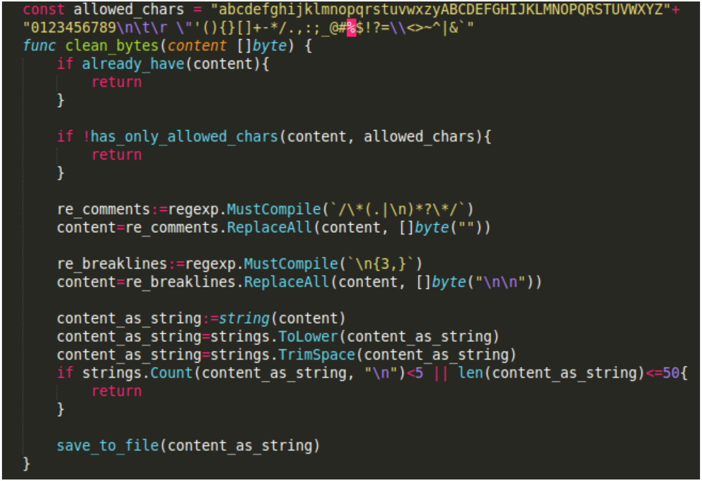
Gathering Data

- Gathering enougth data is often a difficult part of deep learning, because a lot of it is needed.
- I used two training sets:
    - CSS files (4.5 GB of them)
    - A book series (9.3 MB of text)

Gathering Data: CSS Files

- I built a web crawler that would download every css file it finds (except repeated ones).
- Let it run on a NAM computer for a couple of days.
- Actually got a complaint for DOS-like behaviour.

Gathering Data: CSS Files Made Nice



```
const allowed_chars = "abcdefghijklmnopqrstuvwxzyABCDEFGHIJKLMNOPQRSTUVWXYZ"+
"0123456789\n\t\r \"'(){}[]+-*/.,:;_@#%$!?=\\<>~^|&`"
func clean_bytes(content []byte) {
    if already_have(content){
        return
    }

    if !has_only_allowed_chars(content, allowed_chars){
        return
    }

    re_comments:=regexp.MustCompile(`/\*(.|\n)*?\*/`)
    content=re_comments.ReplaceAll(content, []byte(""))

    re_breaklines:=regexp.MustCompile(`\n{3,}`)
    content=re_breaklines.ReplaceAll(content, []byte("\n\n"))

    content_as_string:=string(content)
    content_as_string=strings.ToLower(content_as_string)
    content_as_string=strings.TrimSpace(content_as_string)
    if strings.Count(content_as_string, "\n")<5 || len(content_as_string)<=50{
        return
    }

    save_to_file(content_as_string)
}
```

Gathering Data: Book Series

- I found the books in text format.
- It was an ugly mess translated from scans of the books to text by a computer.
- I removed all the garbage I could, transformed it all to lower case, and removed newline characters.
- In the end only 46 characters were allowed: ”! ’)(-,.103254769;:?acbedgfihkjmlonqpsrutwvyxz”.

Not yet the End

- Show sample input/output.
- Show programs here if there is time.