Recurrent Neural Networks

Emre Kağan Akkaya

N14128491

Outline

- Problem
- Method
  - Definitions
  - Implementation
- Results
- Conclusion

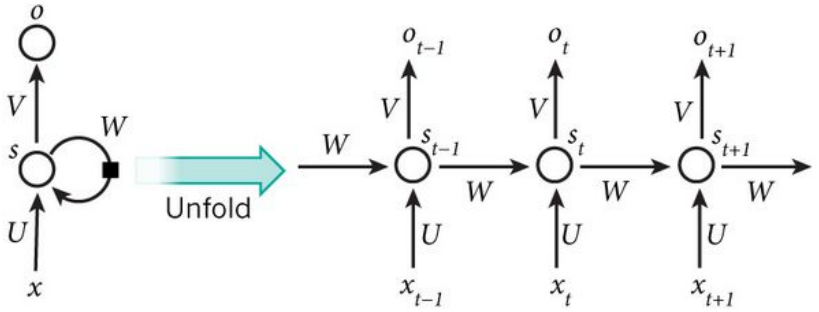
Problem

- Recurrent neural networks (RNN) are powerful & popular models that have attracted great interest and shown promise in many NLP tasks.
  - RNN based language models:
    - Common uses include: speech recognition, machine translation, generating image descriptions etc.
    - Allows to score arbitrary sentences based on how likely they are to occur.
    - Also allows us to generate new text... Why? Because it is fun!
  - Aim of this project is **to use RNN in order to generate Turkish text** based on different training datasets.

What is recurrent neural network?

- In a traditional neural network, it is assumed that all inputs are independent of each other. But if you want to predict the next word in a sentence, we better know which words **came before it**.
  - RNN are recurrent because they perform the same task for every element of the sequence with the output being depended on the previous computations.
  - One can assume they have a **“memory”**...
- RNNs have two sources of input: the present and recent past
  - There is information in the input sequence itself, and RNNs use it to perform tasks that other networks can’t.

How does it work?



- RNN shares the same parameters (*U,V,W*) across all steps
  - Unlike traditional deep neural networks which uses different params at each layer
  - Reduced # of params we need to learn!

x_t is the input at time step *t*.

s_t is the hidden state which is calc. based on the prev. hidden state and the input. $s_t = f(Ux_t + Ws_{t-1})$

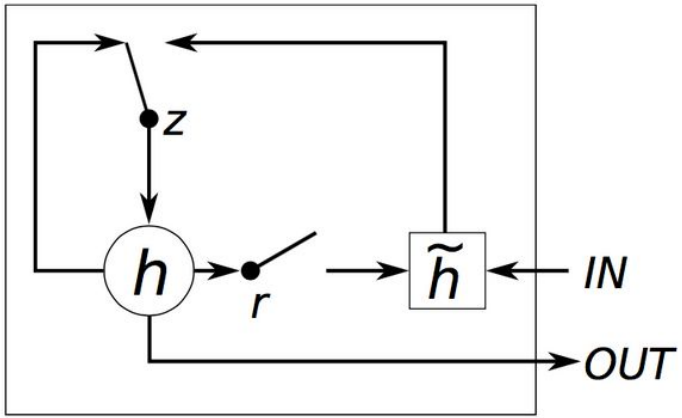
- Training RNNs include backpropagation like others
  - But the gradient at each output depends also on the previous time steps... (Backpropagation through time/**BPTT**)
  - Vanilla RNNs trained with BPTT have difficulties learning long-term dependencies (that are far apart)
    - Vanishing or exploding gradient problem!

LSTM and GRU What are they?

- Our goal is to calculate the gradients of the error (w.r.t *U,V* and *W*) and then learn good parameters using Stochastic Gradient Descent.
  - Gradient values are shrinking exponentially. Eventually vanishing completely…
  - That means those steps doesn't contribute to what we are learning.
  - That's problematic because the meaning of a sentence is often determined by words that aren't very close.
    - Asd
  - **Long Short-Term Memory (LSTM)** and **Gated Recurrent Unit (GRU)** are specifically designed to overcome this problem.
    - Same architecture but different update function to compute the hidden state.
    - In this project, GRU have been implemented.

Gated Recurrent Units

- GRU has two gates, a reset gate *r*, and an update gate *z*.
  - Reset gate determines how to combine the new input with the previous memory
  - Update gate defines how much of the previous memory to keep around.
  - (If we set the *r* to all 1's and *z* to all 0's = vanilla RNN)

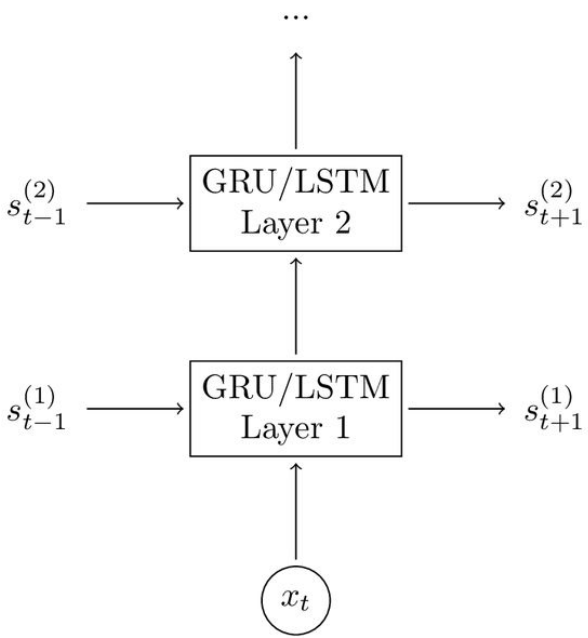


$$z = \sigma(x_t U^z + s_{t-1} W^z)$$
$$r = \sigma(x_t U^r + s_{t-1} W^r)$$
$$h = tanh(x_t U^h + (s_{t-1} \circ r) W^h)$$
$$s_t = (1 - z) \circ h + z \circ s_{t-1}$$

They are called gates because sigmoid function σ squashes the values of vectors between 0 and 1.

Gated Recurrent Units

- Adding a second layer to the network allows our model to capture higher-level interactions
  - But they may lead to overfitting
  - Performance considerations...



Implementation

1. Download lots of Turkish text (Approx. 800MB of Turkish Wikipedia dump and various e-books)
2. Clean text from markup tags, metadata headers etc. (2.195.043 sentences)
3. Tokenize text into words (1.041.577 unique word tokens)
4. Remove infrequent words and build vocabulary with different sizes (2.000, 8.000, 10.000 etc.)
5. Prepend/append *<start>* and *<end>* tokens
6. Build training data matrices
    a. The input to RNNs are vectors, not strings. So we map words to indices. (later, we will use them as one-hot vectors)

Implementation

7. Initialize the parameters *U, V* and *W* to small random values...
8. For each word in the input sentence, forward propagation (GRU, LSTM) make predictions representing the probabilities of the next word.
    a. Predictions are random at first. Need to measure the errors it made! **Cross-entropy loss function** (o_n output predictions, y_n is the correct words)
    b. Remember, the goal is to find *U, V* and *W* that minimize the loss.
        i. Then calc. SGD to nudge the parameters into the right direction
        ii. SGD also needs **learning rate** (how big of a step we should mak with the gradient values calculated by **BPTT**)
        iii. Decrease learning rate if the loss continues to increase!

$$L(y, o) = -\frac{1}{N} \sum_{n \in N} y_n \log o_n$$

9. For each *N* training examples
    a. Generate sentences using model parameters at this step.
    b. Also, save model parameters into a file so that we can use them once we know the model is trained enough.

What did I use?

- Python 2.7
- NLTK library
  - Work tokenization etc.
- Theano
  - To perform operations on GPU
  - Provides efficient implementations for most of the mathematical functions

Simulation environment & parameters

Operating system	Debian GNU/Linux 6 (64bit)
CPU	8 GHz
Memory	16GB RAM memory
Vocabulary size	6.000, 8.000, 10.000
Number of epochs	20, 40
Hidden dimension size	100, 120
Number of GRU layers	2
SGD learning rate	0.001

Results

- Most of the generated text is nonsense...
  - ✓ olarak bir gerekir .
  - ✓ 1967 yılında milli
  - ✓ bir mecmua) .
  - ✓ köyün yeşil adlı sebebiyle tipi takım göç
  - ✓ dolayısıyla ilçesinin bilinir .
  - ✓ milletvekili yataktan dergisinde mektepten
  - ✓ hiçbir oğlundan kendi söylemeyi :
  - ✓ yaptığı doğru herhalde evladım ve sevgili arkasını bende .
  - ✓ anlıyordum söylemişti yeni ?
  - ✓ hayretle bıraktım olsa diye haydi ...
  - ✓ gece buyurun ... dedi . ..
  - ✓ ankara'ya gelirim ... ... tekrar

Results

- ...but some of them are getting meaningful and grammatically correct:
  - ✓ içme suyu şebekesi
  - ✓ uzun 2010 yılında .
  - ✓ köyün ekonomisi tarım ve hayvancılığa dayalıdır .
  - ✓ ptt şubesi ve ptt acentesi yoktur .
  - ✓ 15 mart 2010 tarihinde yapılan bir nüfus sayımına göre şehrin nüfusunun ve
  - ✓ köye ulaşımı sağlayan yol asfalt olup
  - ✓ köyde ilköğretim okulu vardır .
  - ✓ mahallenin adının nereden geldiği ve geçmişi hakkında bilgi yoktur .
- Basically, it just needs parameter fine-tuning & more time to train...

Conclusion

- RNN (specifically *GRU*) **implementation in Python**.
- A **pre-trained language model** which can be used to generate TR text.
  - Still in progress... Training is a long process with this amount of data.
- Turkish datasets or the scripts to generate them
  - Along with the source code, they can be accessed at https://github.com/emrekgn/tr-rnn
- Future work
  - Word2vec or Glove can be used to further optimize results

Thank you for listening!