WILDML

Artificial Intelligence, Deep Learning, and NLP

SEPTEMBER 17, 2015 BY DENNY BRITZ

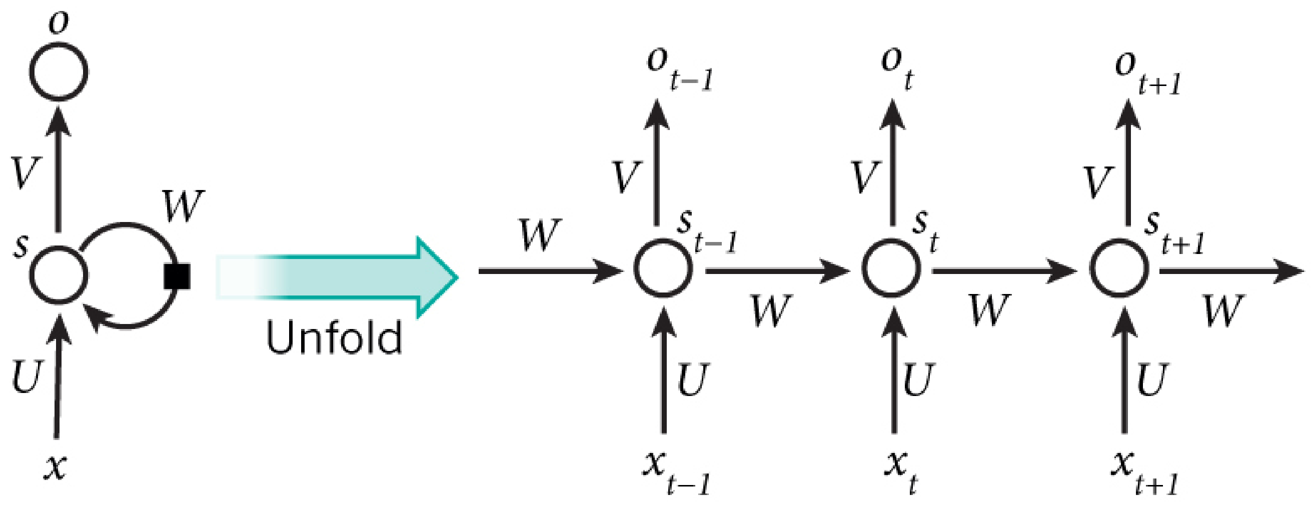
Recurrent Neural Networks (RNNs) are popular models that have shown great promise in many NLP tasks. But despite their recent popularity I've only found a limited number of resources that throughly explain how RNNs work, and how to implement them. That's what this tutorial is about. It's a multi-part series in which I'm planning to cover the following:

1. Introduction to RNNs (this post)
2. Implementing a RNN using Python and Theano
3. Understanding the Backpropagation Through Time (BPTT) algorithm and the vanishing gradient problem
4. Implementing a GRU/LSTM RNN

As part of the tutorial we will implement a recurrent neural network based language model. The applications of language models are two-fold: First, it allows us to score arbitrary sentences based on how likely they are to occur in the real world. This gives us a measure of grammatical and semantic correctness. Such models are typically used as part of Machine Translation systems. Secondly, a language model allows us to generate new text (I think that's the much cooler application). Training a language model on Shakespeare allows us to generate Shakespeare-like text. This fun post by Andrej Karpathy demonstrates what character-level language models based on RNNs are capable of.

I'm assuming that you are somewhat familiar with basic Neural Networks. If you're not, you may want to head over to Implementing A Neural Network From Scratch, which guides you through the ideas and implementation behind non-recurrent networks.

The idea behind RNNs is to make use of sequential information. In a traditional neural network we assume that all inputs (and outputs) are independent of each other. But for many tasks that's a very bad idea. If you want to predict the next word in a sentence you better know which words came before it. RNNs are called because they perform the same task for every element of a sequence, with the output being depended on the previous computations. Another way to think about RNNs is that they have a "memory" which captures information about what has been calculated so far. In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps (more on this later). Here is what a typical RNN looks like:



The above diagram shows a RNN being (or unfolded) into a full network. By unrolling we simply mean that we write out the network for the complete sequence. For example, if the sequence we care about is a sentence of 5 words, the network would be unrolled into a 5-layer neural network, one layer for each word. The formulas that govern the computation happening in a RNN are as follows:

- x_t is the input at time step t. For example, x_1 could be a one-hot vector corresponding to the second word of a sentence.

- s_t is the hidden state at time step t. It's the "memory" of the network. s_t is calculated based on the previous hidden state and the input at the current step: $s_t = f(Ux_t + Ws_{t-1})$. The function f usually is a nonlinearity such as tanh or ReLU. s_{-1}, which is required to calculate the first hidden state, is typically initialized to all zeroes.
- o_t is the output at step t. For example, if we wanted to predict the next word in a sentence it would be a vector of probabilities across our vocabulary. $o_t = \mathrm{softmax}(Vs_t)$.

There are a few things to note here:

- You can think of the hidden state s_t as the memory of the network. s_t captures information about what happened in all the previous time steps. The output at step o_t is calculated solely based on the memory at time t. As briefly mentioned above, it's a bit more complicated in practice because s_t typically can't capture information from too many time steps ago.
- Unlike a traditional deep neural network, which uses different parameters at each layer, a RNN shares the same parameters (U, V, W above) across all steps. This reflects the fact that we are performing the same task at each step, just with different inputs. This greatly reduces the total number of parameters we need to learn.
- The above diagram has outputs at each time step, but depending on the task this may not be necessary. For example, when predicting the sentiment of a sentence we may only care about the final output, not the sentiment after each word. Similarly, we may not need inputs at each time step. The main feature of an RNN is its hidden state, which captures some information about a sequence.

RNNs have shown great success in many NLP tasks. At this point I should mention that the most commonly used type of RNNs are LSTMs, which are much better at capturing long-term dependencies than vanilla RNNs are. But don't worry, LSTMs are essentially the same thing as the RNN we will develop in this tutorial, they just have a different way of computing the hidden state. We'll cover LSTMs in more detail in a later post. Here are some example applications of RNNs in NLP (by non means an exhaustive list).

Language Modeling and Generating Text

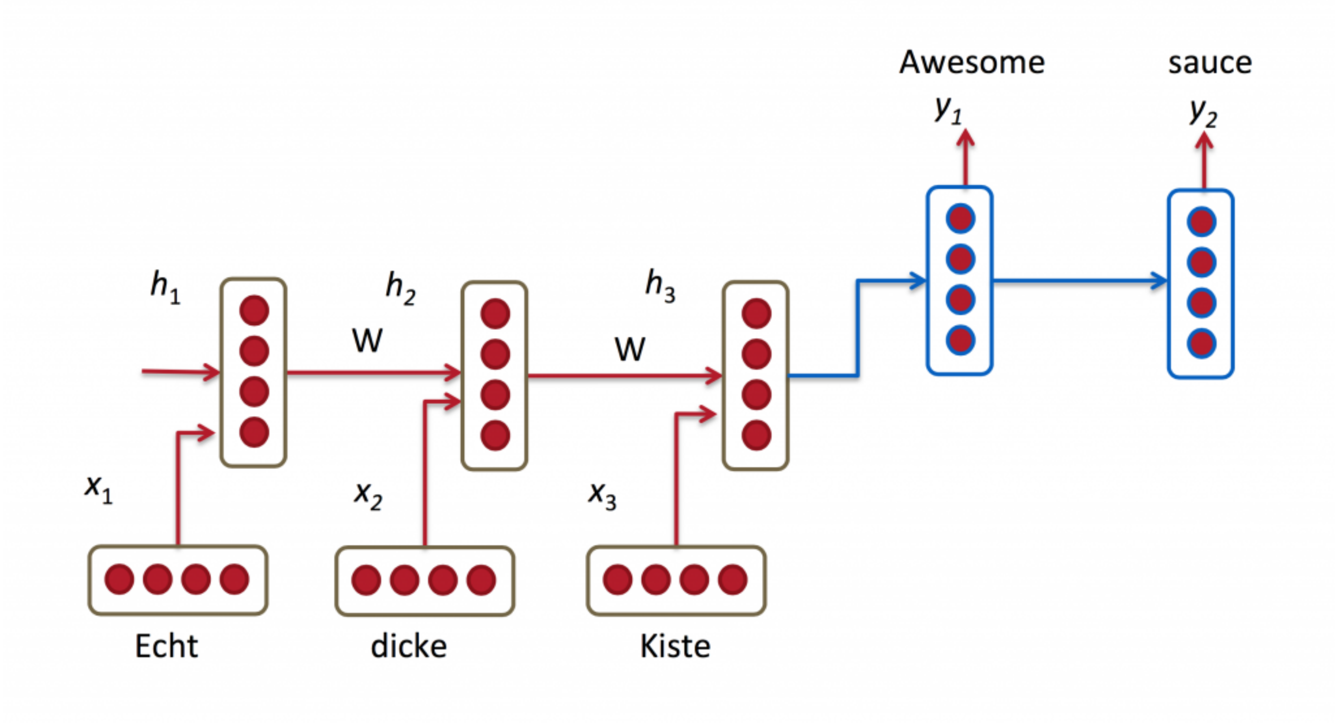
Given a sequence of words we want to predict the probability of each word given the previous words. Language Models allow us to measure how likely a sentence is, which is an important input for Machine Translation (since high-probability sentences are typically correct). A side-effect of being able to predict the next word is that we get a model, which allows us to generate new text by sampling from the output probabilities. And depending on what our training data is we can generate all kinds of stuff. In Language Modeling our input is typically a sequence of words (encoded as one-hot vectors for example), and our output is the sequence of predicted words. When training the network we set $o_t = x_{t+1}$ since we want the output at step t to be the actual next word.

Research papers about Language Modeling and Generating Text:

- Recurrent neural network based language model
- Extensions of Recurrent neural network based language model
- Generating Text with Recurrent Neural Networks

Machine Translation

Machine Translation is similar to language modeling in that our input is a sequence of words in our source language (e.g. German). We want to output a sequence of words in our target language (e.g. English). A key difference is that our output only starts after we have seen the complete input, because the first word of our translated sentences may require information captured from the complete input sequence.



Research papers about Machine Translation:

- A Recursive Recurrent Neural Network for Statistical Machine Translation
- Sequence to Sequence Learning with Neural Networks
- Joint Language and Translation Modeling with Recurrent Neural Networks

Speech Recognition

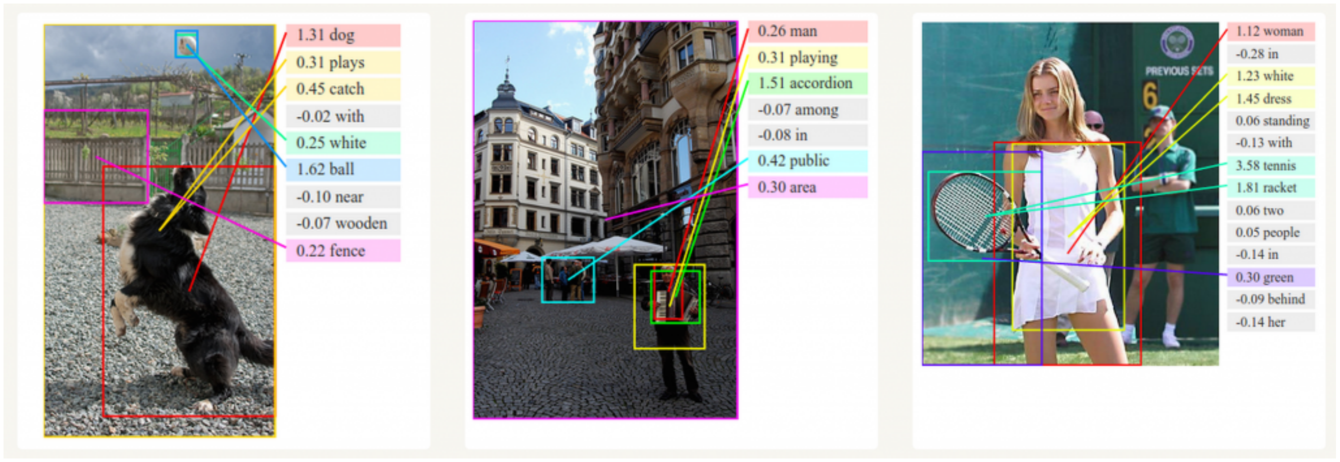
Given an input sequence of acoustic signals from a sound wave, we can predict a sequence of phonetic segments together with their probabilities.

Research papers about Speech Recognition:

- Towards End-to-End Speech Recognition with Recurrent Neural Networks

Generating Image Descriptions

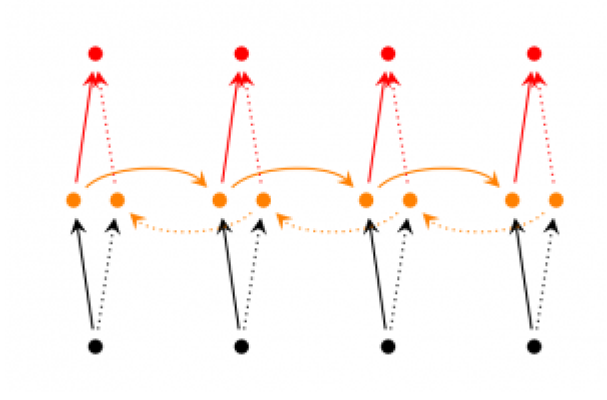
Together with convolutional Neural Networks, RNNs have been used as part of a model to generate descriptions for unlabeled images. It's quite amazing how well this seems to work. The combined model even aligns the generated words with features found in the images.



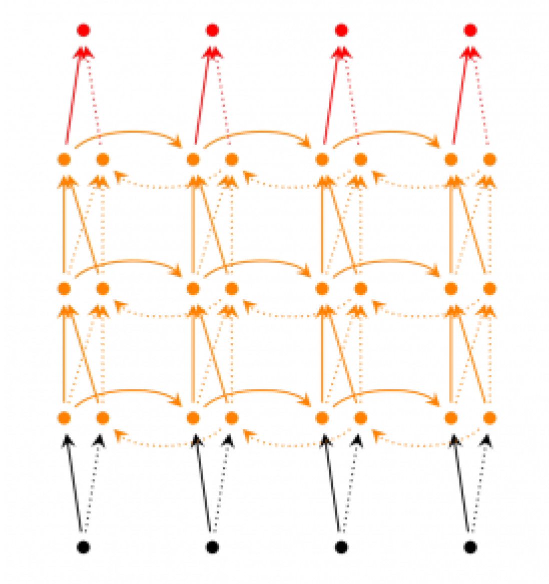
Training a RNN is similar to training a traditional Neural Network. We also use the backpropagation algorithm, but with a little twist. Because the parameters are shared by all time steps in the network, the gradient at each output depends not only on the calculations of the current time step, but also the previous time steps. For example, in order to calculate the gradient at $t = 4$ we would need to backpropagate 3 steps and sum up the gradients. This is called Backpropagation Through Time (BPTT). If this doesn't make a whole lot of sense yet, don't worry, we'll have a whole post on the gory details. For now, just be aware of the fact that vanilla RNNs trained with BPTT have difficulties learning long-term dependencies (e.g. dependencies between steps that are far apart) due to what is called the vanishing/exploding gradient problem. There exists some machinery to deal with these problems, and certain types of RNNs (like LSTMs) were specifically designed to get around them.

Over the years researchers have developed more sophisticated types of RNNs to deal with some of the shortcomings of the vanilla RNN model. We will cover them in more detail in a later post, but I want this section to serve as a brief overview so that you are familiar with the taxonomy of models.

Bidirectional RNNs are based on the idea that the output at time t may not only depend on the previous elements in the sequence, but also future elements. For example, to predict a missing word in a sequence you want to look at both the left and the right context. Bidirectional RNNs are quite simple. They are just two RNNs stacked on top of each other. The output is then computed based on the hidden state of both RNNs.



Deep (Bidirectional) RNNs are similar to Bidirectional RNNs, only that we now have multiple layers per time step. In practice this gives us a higher learning capacity (but we also need a lot of training data).



LSTM networks are quite popular these days and we briefly talked about them above. LSTMs don't have a fundamentally different architecture from RNNs, but they use a different function to compute the hidden state. The memory in LSTMs are called and you can think of them as black boxes that take as input the previous state h_{t-1} and current input x_t. Internally these cells decide what to keep in (and what to erase from) memory. They then combine the previous state, the current memory, and the input. It turns out that these types of units are very efficient at capturing long-term dependencies. LSTMs can be quite confusing in the beginning but if you're interested in learning more this post has an excellent explanation.

So far so good. I hope you've gotten a basic understanding of what RNNs are and what they can do. In the next post we'll implement a first version of our language model RNN using Python and Theano. Please leave questions in the comments!

DEEP LEARNING, NEURAL NETWORKS, RECURRENT NEURAL NETWORKS

93 Comments **WildML** 1 **Login**

Recommend 36 Tweet Share Sort by Best

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

mina khoshdeli • 3 years ago

Thank you, it is great!

60 • Reply • Share ›

Michael David Watson • 4 years ago

So what exactly are U, V and W? You mention that they are the same parameters for each step, and that state and output are dependent on them, but I am not sure if they are just place holders for example paramaters or have a specific meaning.

9 • Reply • Share ›

Denny Britz Mod → Michael David Watson • 4 years ago • edited

U, V and W are the parameters of the network that you need to learn from your training data. So before you've trained your network using data you don't usually know the "content" of these matrices.

Here's an example for Language Modeling for when you have 5 words in your vocabulary: I, am, the, super, man.

- Your input vectors x would be of dimension 5, essentially selecting a word, e.g. [0 0 1 0 0] ("the")

- Together U and W define how to calculate the new memory of the network given the previous memory and the input word (together with the function f). You start out with randomly initialized U and W matrices, but as you train your network U will learn how to map the word vector above into the "space" of the hidden layer, so it would be of size [D]x5, and after multiplication with x you get a result of size [D]. You can pick [D], it's how big you want the hidden state state to be.

- V defines how to map the hidden state (memory) back into the space of possible words, so it's of size 5x[D]. Multiplying V and s would give you a vector of scores for the predicted word, e.g. [2313, 31, 11, 55, 113] and the softmax would convert these scores into probabilities. The word with the highest probability (the first element in the above example) would then become your prediction.

Does that make sense?

6 ^ | ⌄ • Reply • Share ›

skotadi ➦ Denny Britz • 4 years ago

So U is the "Word Vector to Hidden Space map", and V is the "Hidden Space to Word Vector map." What would a good name for W be in this context?

2 ^ | ⌄ • Reply • Share ›

Show more replies

Show more replies

Zozozoz • 4 years ago

Great post for me, can't wait for the next one.

3 ^ | ⌄ • Reply • Share ›

Nick Byrne • 4 years ago

Thanks for a great intro to RNNs, I look forward to the following ones.
Have you seen any good 'side by side' comparisons of *NN algorithms performance?

2 ^ | ⌄ • Reply • Share ›

Denny Britz Mod ➦ Nick Byrne • 4 years ago • edited

Thanks Nick! It's bit difficult to compare *NN architectures because many of them solve fundamentally different problems. For example, you can't use traditional Deep Neural Networks to solve the problems that RNNs solve. I haven't seen any papers that do a detailed comparison, but there probably are some, I just don't know about them. A quick Google search turned up this (relatively new) one:

http://jmlr.org/proceedings... - It compares several architectures (mostly LSTMs) across wide range of hyperparameters.

Edit: Also, I just remembered this one (again, mostly concerned with LSTMs): http://arxiv.org/pdf/1503.0...

1 ^ | ⌄ • Reply • Share ›

Mattias Fagerlund ➦ Denny Britz • 2 years ago

Regarding "DNNs can't solve the problems RNNs solve"; this used to be consideredd true, but WaveNet (https://deepmind.com/blog/w...) demonstrate how DNNs (through convolutions over time) actually can do what RNNs do - in some cases...

cheers,
m

^ | ⌄ • Reply • Share ›

Show more replies

Jijitsu • 3 years ago

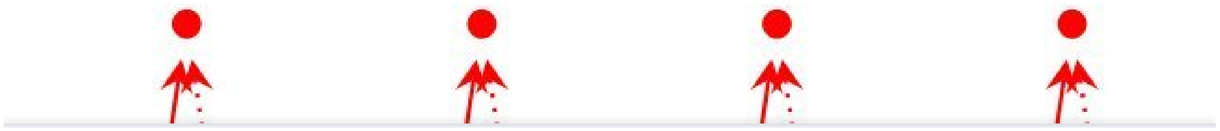
First of all thanks for the great tutorial. I have a couple of questions:

"x_t is the input at time step t. For example, x_1 could be a one-hot vector corresponding to the second word of a sentence." So the first word of a sentence would be x_0?

"When training the network we set o_t = x_{t+1} since we want the output at step t to be the actual next word." I don't understand this statement. Why equate an output directly with the next input, ignoring all the hidden layers? Shouldn't the output o_t be dependent on the hidden layers even when training the network?

In the section where you explain a taxonomy of models you have a figure. Could you please clarify what each colour is? Is it Black: input, Orange: Hidden Layer, Red: Output?

Thanks in advance.



see more

1 ^ | ⌄ • Reply • Share ›

Denny Britz Mod ➦ Jijitsu • 3 years ago

Hi Jubei,

> So the first word of a sentence would be x_0

Yes, that's right.

> I don't understand this statement. Why equate an output directly with the next input, ignoring all the hidden layers? Shouldn't the output o_t be dependent on the hidden layers even when training the network?

Yes, you're right. The output *is* dependent on the hidden layers, but you need a supervision signal that tells the network what would be the correct output would be. The hidden layers compute the output, but you're using the actual next word to train the network. Does that make sense?

> In the section where you explain a taxonomy of models you have a figure. Could you please clarify what each colour is? Is it Black: input, Orange: Hidden Layer, Red: Output?

Yes, that's right. Black: input, Orange: Hidden Layer, Red: Output.

^ | ˅ • Reply • Share ›

Jijitsu ➦ Denny Britz • 3 years ago

>you're using the actual next word to train the network

Yes it makes sense now. Thank you.

^ | ˅ • Reply • Share ›

Martyn Mlostekk • 4 years ago • edited

Great introduction, cant wait for the next part!

1 ^ | ˅ • Reply • Share ›

Denny Britz Mod ➦ Martyn Mlostekk • 4 years ago

Thanks Martyn, glad to hear that!

^ | ˅ • Reply • Share ›

Mark Wissler • 4 years ago

This is a great post. Very exciting for the next one.

1 ^ | ˅ • Reply • Share ›

Denny Britz Mod ➦ Mark Wissler • 4 years ago

Thanks Mark! Glad you liked it!

^ | ˅ • Reply • Share ›

William Davis • 4 years ago

Wow. Great article. Cant wait for the next post. This is an area of CS I've not had much exposure, but after reading this I'm geeking out a little.

1 ^ | ˅ • Reply • Share ›

Denny Britz Mod ➦ William Davis • 4 years ago

Glad to hear that! And thanks for reading :)

^ | ⌄ • Reply • Share ›

Suresh Kumar • 3 months ago

Great article **@Denny Britz**, it gave very good understanding of RNNs and its applications. Please keep up the good work.:-)

^ | ⌄ • Reply • Share ›

Фаиль Гафаров • 3 months ago

Nice post! Thank you! Users can learn about recurrent neural networks from another blogs too. For example
https://learn-neural-networ...
LSTM neural networks: https://learn-neural-networ...

^ | ⌄ • Reply • Share ›

Mihir Gajera • 4 months ago

what are the dimensions of U,V,W and St? (Generalised dimensions).

^ | ⌄ • Reply • Share ›

Shivam Gupta • 9 months ago

Hello Guys I was working on the Building, Tank Prediction on the Satellite Images. I have Implemented with the combination of ResNEt and RefineNet, But I am not getting the required results, I analyzed Via PCA that the Outputs just before the Last Classifier Block is Giving Good Results(Features are able to distinghuish between the tank, building an background) but the Classifier Block is Not giving Good Results on the Basis of Conv2D with Logits . So I was trying to Change the Classifier Block, I was trying to go for SVM at the end of DNN. Could anyone Help with how to decide the classifier block and what could be the best classifier at the end of DNN. Thanks in Advance.

^ | ⌄ • Reply • Share ›

dong xiao • 10 months ago

Thank you ,very good

^ | ⌄ • Reply • Share ›

张杰 • a year ago

Nice job! This gives me a lot of help. Thank you very much.

^ | ⌄ • Reply • Share ›

樂以虎 • a year ago

Great!

^ | ⌄ • Reply • Share ›

candy • a year ago

Thank you so much for the sharing.
I have a question. How do we determine the starting and ending of a training sample? For example if we are feeding in a very very long passage of essay to train it to deduce the next word, what would be the starting and ending of each training sample?

next word, what would be the starting and ending of each training sample?

Taking an example from this sentence:
.......Peter live in France. He likes to eat bread. He also speak fluent French........

In the above example, in order to deduce the word French, the country France will have to be taken into account. Then how would I know that my starting point should be "Peter "and ends at "French"? Especially if we are given a very long passage, I have no idea how to go about doing it.
I have googled that vanilla RNN cannot support very long chain and that is why they suggest to use LSTM. Does it mean LSTM chain can go endless? But it does not quite make sense to me as we still need to do back propagation, so the chain should not be endless.

How do we then determine how long the chain should be?

• Reply • Share ›

Yassine Kardid • a year ago

hello sir, I'm working on a project in the industrial field whose goal is to detect pebbles in a product in the production chain by analyzing the images taken by the surveillance cameras. I did a lot of research I found that there is a lot of algorithm that can be useful like CNN, SVM, KNN, RNN.
Please how can I know the algorithm that will work properly in this case ?

• Reply • Share ›

Thanaphong Joe Phongpreecha • a year ago

Such an excellent explanation

• Reply • Share ›

FlightOfFancyBee • 2 years ago • edited

Quick question: shouldn't the unfolding be done by connecting s_{t-1} to o_t instead of o_{t-1} ? Just like W takes one step, so should V. In this way the diagram would need to have the Vs pointing diagonally to the right. (The same with Us it seems).

Just curious, I'm not an expert, but wanted to clarify.

• Reply • Share ›

General Useage • 2 years ago

Amazing high level overview thank you for this!

• Reply • Share ›

钟钟 • 2 years ago

Vanilla RNN? I searched it in google, but still couldn't get its clear definition. Sorry, English is not my native language. What's its true meaning?

• Reply • Share ›

钟钟 ➦ 钟钟 • 2 years ago

I searched Vanilla meaning in oxford dictionary. This meaning is correct?

ADJECTIVE
informal
Having no special or extra features; ordinary or standard.

• Reply • Share ›

Alexander Yau ➜ 钟钟 • 2 years ago • edited

If you describe a person or thing as vanilla, you mean that they are ordinary, with no special or extra features. So vanilla RNN is the most ordinary RNN, it is the most common and simple RNN

1 • Reply • Share ›

Show more replies

Dipti Mishra • 2 years ago

Hi,I am not able to understand ,how image compression is taking place,through RNN?

• Reply • Share ›

maritn • 2 years ago

excellent post

• Reply • Share ›

Suyash Gupta • 2 years ago

hi Denny,could u please provide me with the code for chunking using bi lstm...?

• Reply • Share ›

Shlomi Schwartz • 2 years ago

How would you go implementing fraud detection using RNN's? For example, given a time series data finding the anomalies.

• Reply • Share ›

karan patel • 2 years ago

How many types of neural networks are there still now that we can use or know ?

• Reply • Share ›

Trang Thieu Gia Nsc • 2 years ago

Can you tell me about timestep of LSTM in keras mean? and with example: [[1,2,3,4], [2,3,4,5],..[1,2,3,5]] with dimension just (n,4); and output is [0,1,0,1,...0] or with version of 1 hot encoding: [[1,0],[0,1], ..[0,1]] how can i use LSTM layer recevie input layer is 3 Dimension with time_step != 1. I can reshape this input but i want to if i reshape input to another timestep so output need reshape too or something?

• Reply • Share ›

kishore p v • 2 years ago • edited

You mention that "LSTMs are called cells and you can think of them as black boxes that take as input the previous state h(t-1) and current input x(t)". But, in the post http://colah.github.io/post... we see that the state is C(t), and h(t) is the "filtered version" of the state.

• Reply • Share ›

Cheng-Wei Wu • 2 years ago

Amazing post!!!!! It's very helpful for me to have a basic view of RNN:)
Thank u very muchhhhhh!

• Reply • Share ›

Green • 2 years ago

I just do not understand, s and o are vectors or scalar quantities (vector size = 1)? I have read and found that there are more than one way to implement a RNN. Each neuron in the hidden layer has a feedback itself or the whole hidden layer has a feedback. Which model do you mention in this article?

• Reply • Share ›

SeanBearden • 2 years ago

Thanks for the time and effort you put into this post!

• Reply • Share ›

AhmedESamy • 2 years ago • edited

You mentioned before that many of NN architectures solve fundamentally different problems. For example, you can't use traditional Deep Neural Networks to solve the problems that RNNs solve. So what are the fundamental differences between the NN and RNN in terms of the type of problems they solve? more details and reasons are better.

• Reply • Share ›

Anubhav Gupta • 2 years ago

Very well written post. Link to references is also very useful.

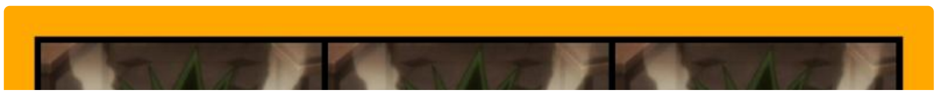
• Reply • Share ›

isaac kargar • 2 years ago

Hi,
Tnx for your great blog. I have a question. when we want to predict words in a sentence we should convert one word to a one hot vector. dimension of this vector is as our dictionary (n) . at step t we should apply this one hot vector to rnn? I really confused about number of neurons in the network. and at step t+1 should we apply the next one hot vector(word) to rnn? to diffrent neurons? can we apply a sentence (a couple of words at step t or not? and for multi layer rnn every layer will continue as we go ahead through time?!

• Reply • Share ›

Achraf Benlemkaddem • 3 years ago



Reply • Share ›

Sandrine • 3 years ago

Thx for introduction to the RNN, am asking how can i use it to generate Shakespeare's