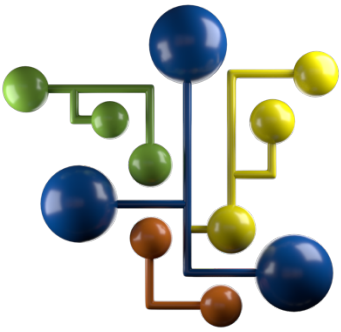


Data Science Academy

www.datascienceacademy.com.br

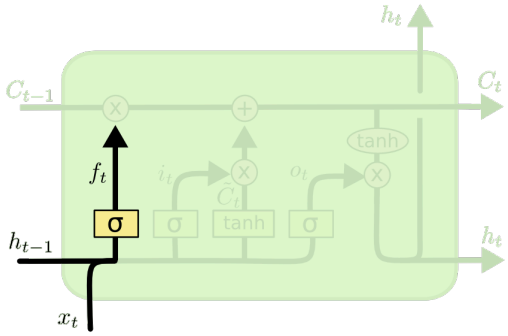
Processamento de Linguagem Natural

Como Funcionam as LSTMs

Vamos compreender em mais detalhes, como funcionam as LSTMs e na sequência avaliar algumas de suas variações, como as GRUs.

O primeiro passo no modelo LSTM é decidir quais as informações que vamos “afastar” da célula de estado. Essa decisão é feita por uma camada sigmoide chamada "forget gate layer" (ou Camada do Esquecimento). Esta camada recebe ht - 1 e Xt e emite um output entre 0 e 1 para cada número na célula de estado Ct - 1. O valor 1 representa "manter completamente isso" enquanto um 0 representa "se livrar completamente disso".

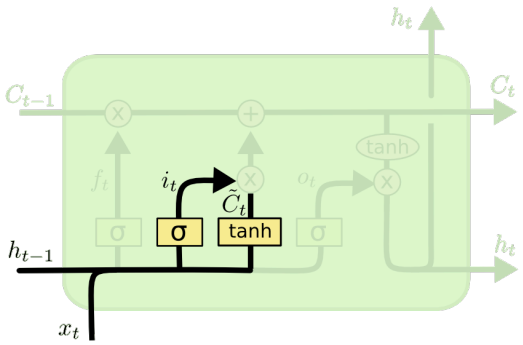
Vejamos o exemplo de um modelo de linguagem tentando prever a próxima palavra com base em todas as anteriores. Em tal problema, a célula de estado pode incluir o gênero do assunto atual, para que os pronomes corretos possam ser usados. Quando vemos um novo assunto, queremos esquecer o gênero do assunto antigo.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

O próximo passo é decidir quais novas informações vamos armazenar na célula de estado. Isso tem duas partes. Primeiro, uma camada sigmoide chamada "Input Gate Layer" (ou Portão de Entrada) que decide quais valores vamos atualizar. Em seguida, uma camada tanh cria um vetor de novos valores candidatos, Ct, que podem ser adicionados ao estado. No próximo passo, combinaremos estes dois para criar uma atualização para o estado.

No exemplo do nosso modelo de linguagem, gostaríamos de adicionar o gênero do novo assunto à célula de estado, para substituir o antigo que estamos nos esquecendo.



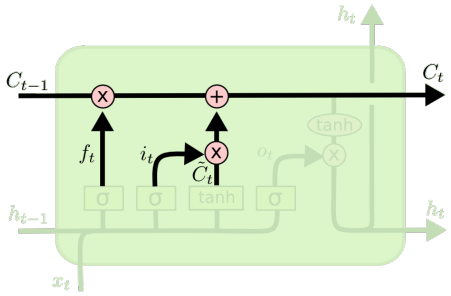
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Agora é hora de atualizar a célula de estado anterior, Ct - 1, na nova célula de estado Ct. As etapas anteriores já decidiram o que fazer, só precisamos realmente fazê-lo.

Nós multiplicamos o estado antigo por f_t, esquecendo as coisas que decidimos esquecer anteriormente. Então, adicionamos $i_t * \tilde{C}_t$. Estes são os novos valores candidatos, escalados em relação ao quanto decidimos atualizar cada valor do estado.

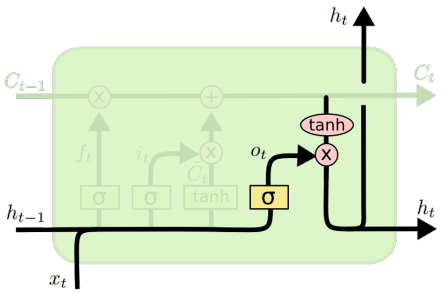
No caso do modelo de linguagem, é aqui que realmente deixamos a informação sobre o gênero do sujeito antigo e adicionamos as novas informações, conforme decidimos nas etapas anteriores.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Finalmente, precisamos decidir o que vamos gerar como output. Esta saída será baseada em nossa célula de estado, mas será uma versão filtrada. Primeiro, nós executamos uma camada sigmoid que decide quais partes da célula de estado serão usadas no output. Em seguida, colocamos a célula de estado através da função de ativação tanh (para que os valores estejam entre -1 e 1) e multiplicamos pela saída do portão sigmoid, de modo que nós apenas produzimos as partes que decidimos.

Para o exemplo do modelo de idioma, uma vez que apenas vimos um assunto, podemos querer exibir informações relevantes para um verbo, caso seja o que vem depois. Por exemplo, podemos produzir se o sujeito é singular ou plural, para que possamos saber em que forma um verbo deve ser conjugado se for o que vem a seguir.

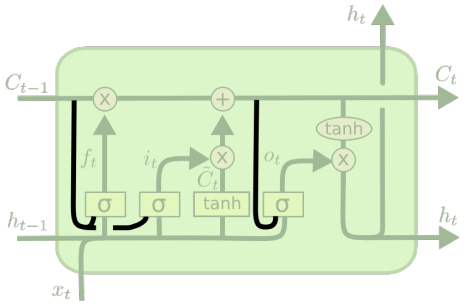


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Variantes na Memória de Longo Prazo Longo

O que descrevemos até agora é um modelo LSTM bastante normal. Mas nem todos os LSTMs são os mesmos conforme descrito acima. Na verdade, parece que quase todos os trabalhos envolvendo LSTMs usam uma versão ligeiramente diferente. As diferenças são poucas, mas vale a pena mencionar algumas delas.

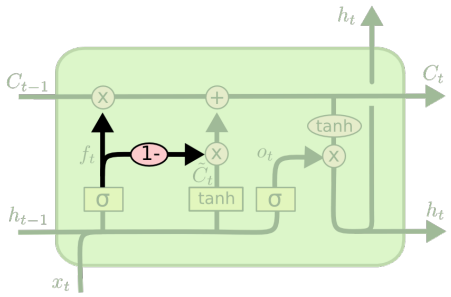
Uma variante LSTM popular, introduzida por Gers & Schmidhuber (2000), adiciona "conexões peephole". Isso significa que deixamos as camadas do gate olhar para o estado da célula.



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

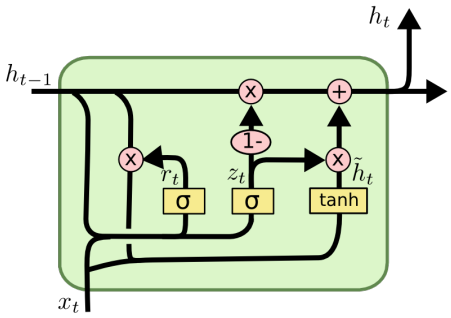
O diagrama acima adiciona *peepholes* a todos os portões.

Outra variação é usar coupled forget e input gates. Em vez de decidir separadamente sobre o que esquecer e sobre o que devemos adicionar novas informações, juntamos essas decisões. Nós só esquecemos quando entramos algo em seu lugar. Somamos apenas novos valores para o estado quando esquecemos algo mais antigo.



$$C_t = f_t * C_{t-1} + (\mathbf{1} - \boldsymbol{f_t}) * \tilde{C}_t$$

Uma variação ligeiramente mais dramática no modelo LSTM é a Gated Recurrent Unit, ou GRU, introduzida por Cho, et al. (2014). Combina os portões de esquecimento e de entrada em um único "portão de atualização". Ele também mescla o estado da célula e o estado oculto e faz algumas outras mudanças. O modelo resultante é mais simples do que os modelos LSTM padrão e tem crescido cada vez mais em popularidade.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Estas são apenas algumas das variantes LSTM mais notáveis. Há muitos outros, como Deep RNN Gated por Yao, et al. (2015). Há também uma abordagem completamente diferente para lidar com dependências de longo prazo, como RNN Clockwork por Koutnik, et al. (2014).

Qual dessas variantes é melhor? As diferenças são importantes? Greff, et al. (2015) fazem uma boa comparação de variantes populares, achando que elas são quase as mesmas. Jozefowicz, et al. (2015) testaram mais de dez mil arquiteturas RNN, encontrando algumas que funcionaram melhor do que LSTMs clássicas em certas tarefas.

Referências:

Recurrent Nets that Time and Count
ftp://ftp.idsia.ch/pub/juergen/TimeCount-IJCNN2000.pdf

A Clockwork RNN
https://arxiv.org/pdf/1402.3511v1.pdf

Depth-Gated RNN
https://arxiv.org/pdf/1508.03790v2.pdf

LSTM: A Search Space Odyssey
https://arxiv.org/pdf/1503.04069.pdf

An Empirical Exploration of RNN Architecture
http://proceedings.mlr.press/v37/jozefowicz15.pdf