TSOLAS LEONIDAS
MSc in Data Science - AUEB
DEEP LEARNING

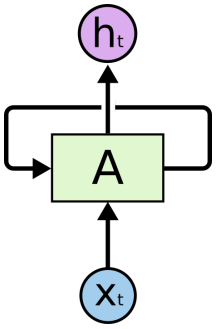
Introduction

The purpose of this project is to predict different types of toxicity like threats, insults and identity-based hate among people interacting in social networking. Wikipedia provides a dataset which contains comments posted by users which have been labeled in terms of toxicity by users. A single comment may have multiple toxicity levels at the same time . More specifically, the six categories of toxicity are *toxic, severe_toxic, obscene, threat, insult, identity_hate.* In order to identify which kinds of toxicity are present in a comment, if any, a Deep Learning Model is built and evaluated.

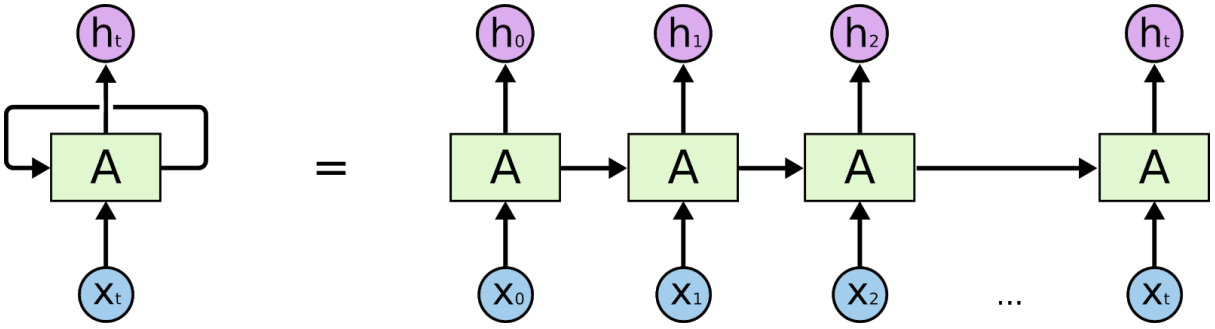
Deep Learning Approach

RNNs

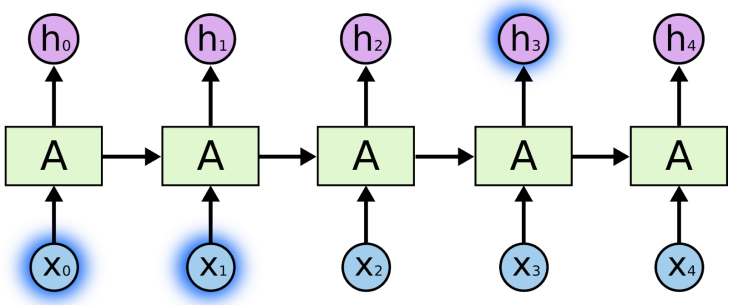
The principle here is that the meaning of each comment is based on understanding previous words. Recurrent neural networks (RNNs) are networks with loops in them, allowing information to persist.



An RNN can be faced as multiple copies of the same network, each one passing a message to a successor.

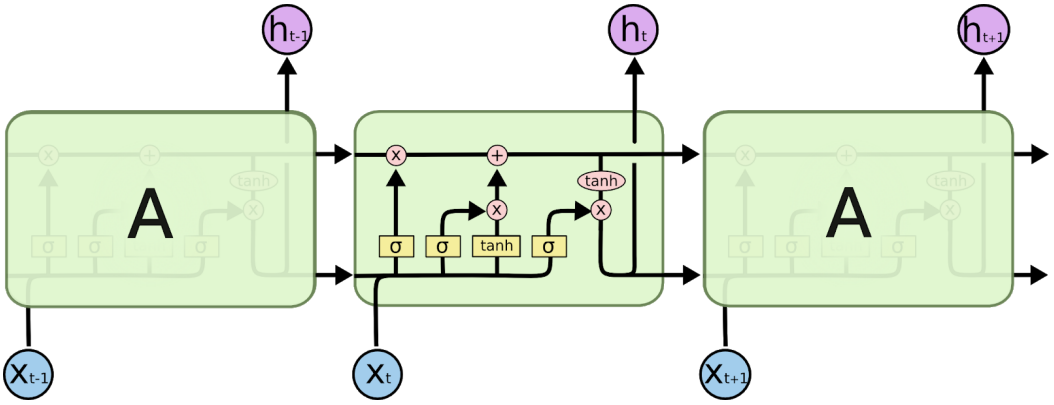


One of the main ideas of RNNs is that they are able to connect previous information to the present task, in our case connecting the context of words with previous ones in the sequence of comments.

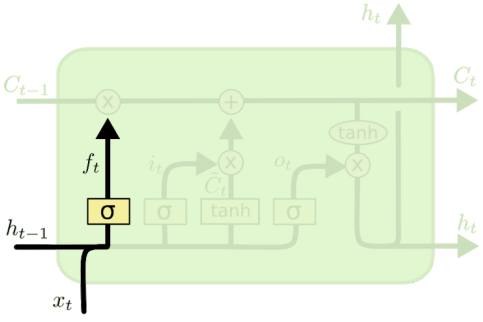


Long Short Term Memory networks (LSTMs) are a special kind of RNN, capable of learning long-term dependencies and are explicitly designed to avoid the long-term dependency problem.

Structure of LSTM

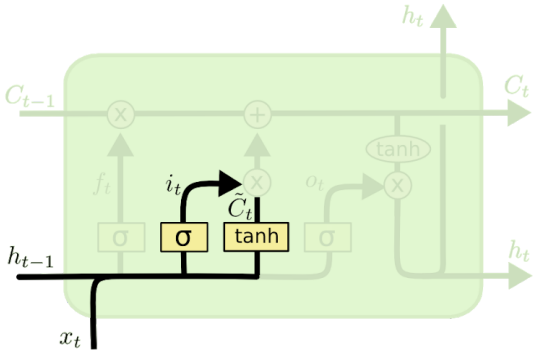


The first step in our LSTM is to decide what information should be thrown away from the cell state. This decision is made by a sigmoid layer called the **forget gate layer**.



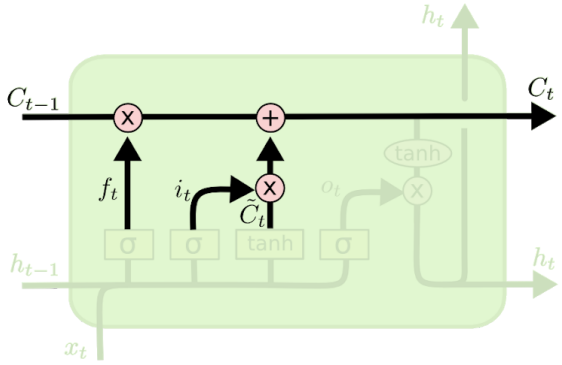
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

The next step is to decide what new-added information is going to be stored in the cell state. A sigmoid layer called the **input gate layer** decides which values will be updated. A tanh layer creates a vector of new values that will be added to the cell state.



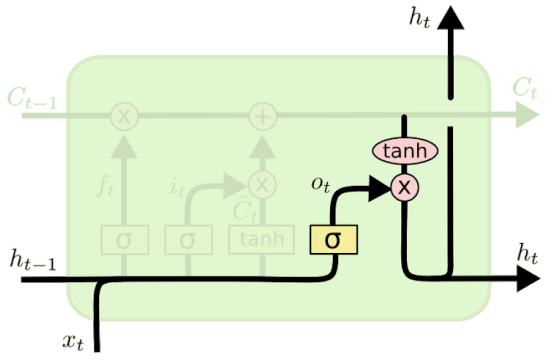
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

It's now time to update the cell state by adding the previous information to the new one.



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

In order to get the output we run a sigmoid layer which decides what parts of the cell state are pushed to output. Then, the cell state passes through tanh to push the values between −1 and 1 and multiply it by the output of the sigmoid gate.



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

Methodology

1. Check for missing values
2. Tokenize comments into unique words
3. Index words using a dictionary
4. Define the number of unique words in the dictionary as MAXFEATURES
5. Use padding for comments in order to get fixed length comments into batch
6. Build an LSTM Model:
    a. Input: List of encoded sentences of dimension MAXLENGTH
    b. Embedding layer gives list of coordinates of words in vector space of dimension EMBEDSIZE

c. LSTM produces output of dimension 60
   i. The unrolled sequence of results is passed from each recursion to the next layer
   ii. Output of Embedding layer is a 3D Tensor (None,MAXLENGTH,EMBEDSIZE)
   iii. Reshape into 2D Tensor using GlobalMaxPool and pass into LSTM
   iv. Set Dropout rate output of dimension DROPOUT to disable nodes and enhance generalization
   v. Use of Sigmoid function to achieve binary classification for each label
   vi. Loss function is optimized using Adam optimizer
   vii. Binary CrossEntropy is defined as loss function
   viii. Number of EPOCHS is defined
   ix. Run Model

	MAX FEATURES	MAX LENGTH	EMBED SIZE	EPOCHS	DROPOUT	VALIDATION SPLIT	DENSE	Accuracy %	Validation Accuracy %	Time (sec)
1	20000	223	128	2	0.10	0.1	50	98.30	98.28	771
2	20000	223	128	2	0.10	0.1	50	98.29	98.22	864
3	20000	223	128	2	0.10	0.2	50	98.61	98.18	599
4	20000	148	128	4	0.10	0.2	50	98.31	98.24	1581
5	20000	148	128	2	0.15	0.2	50	98.30	98.25	584
6	20000	148	128	2	0.10	0.2	50	98.28	98.27	581
7	20000	148	128	2	0.10	0.2	50	98.25	98.24	588
8	15000	148	128	2	0.10	0.2	50	98.25	98.20	534
9	20000	148	256	2	0.10	0.2		98.31	98.15	919
10	20000	148	128	2	0.10	0.2	60	98.29	98.23	598

The version 6 of the executions would be chosen. This comes from the best balance among the others in terms of accuracy in validation test and the total time of execution.