CIS 8045

Word Embedding & RNN

TF-IDF

- Inverse Document Frequency (IDF)

$$idf(t) = \log [n \ / \ df(t)] + 1$$

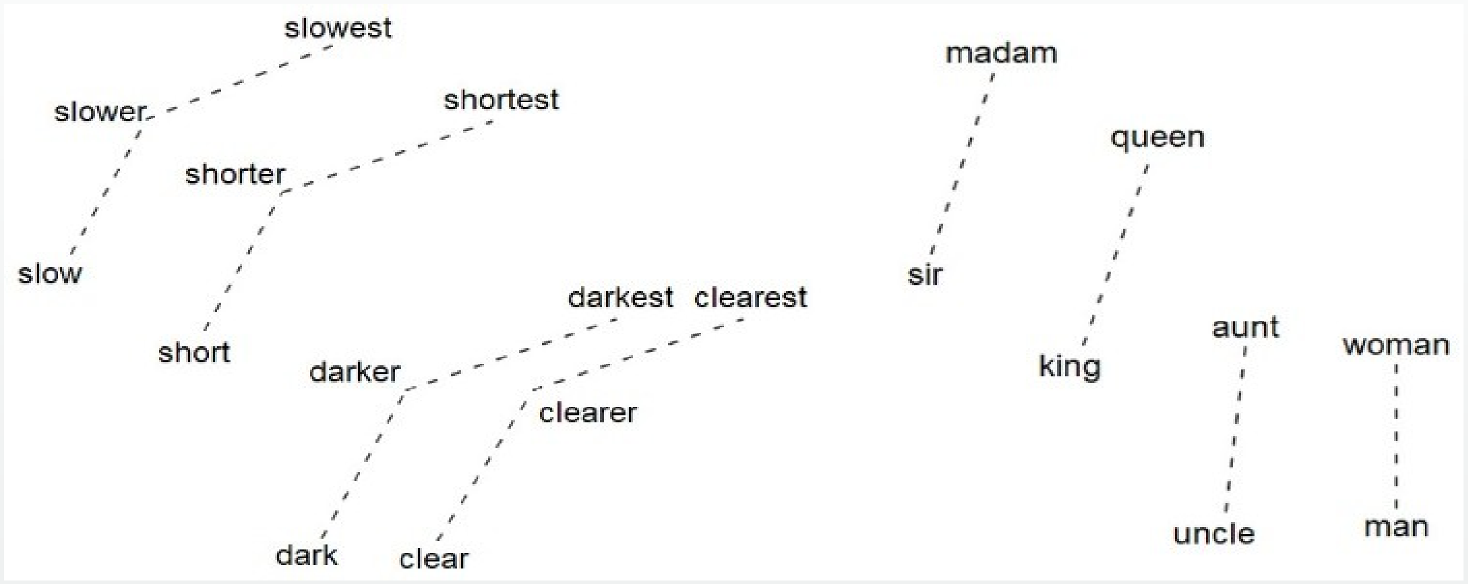
n is the total number of documents, while *df(t)* is the number of documents where the term t occurs. This is used as a factor to adjust the term frequency. It increases the importance for rare terms, and decreases it for common terms.

Word Embedding

Guess the meaning of the term ***furbaby*** from the following text:

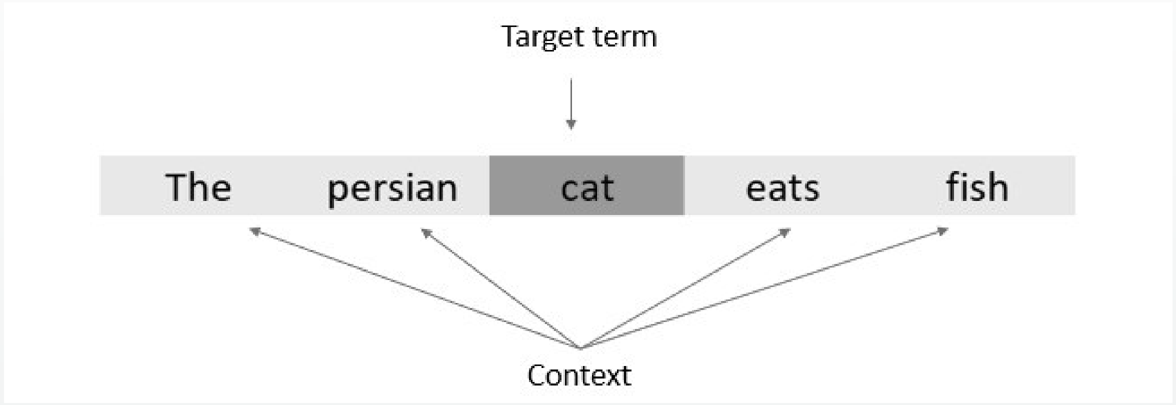
"I adopted a young Persian furbaby a month back. Like all furbabys, it loves to scratch its back and hates water, but unlike other furbabys, it miserably fails at catching a mouse."

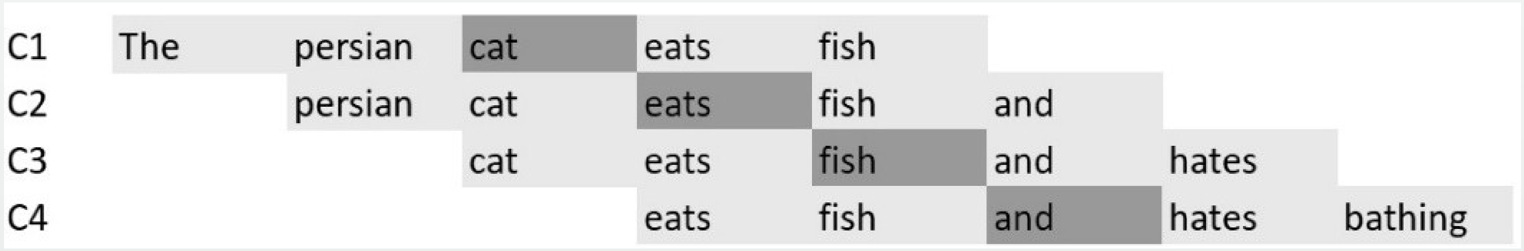
Word Embedding



the segment connecting **slow** and **slower** is parallel to the segment connecting **short** and **shorter**. This means that the word embeddings learned that the relationship between **short** and **shorter** is the same as the relationship between **slow** and **slower**. Likewise, the embeddings learned that the relationship between **clearer** and **clearest** is the same as that between **darker** and **darkest**.

word2vec

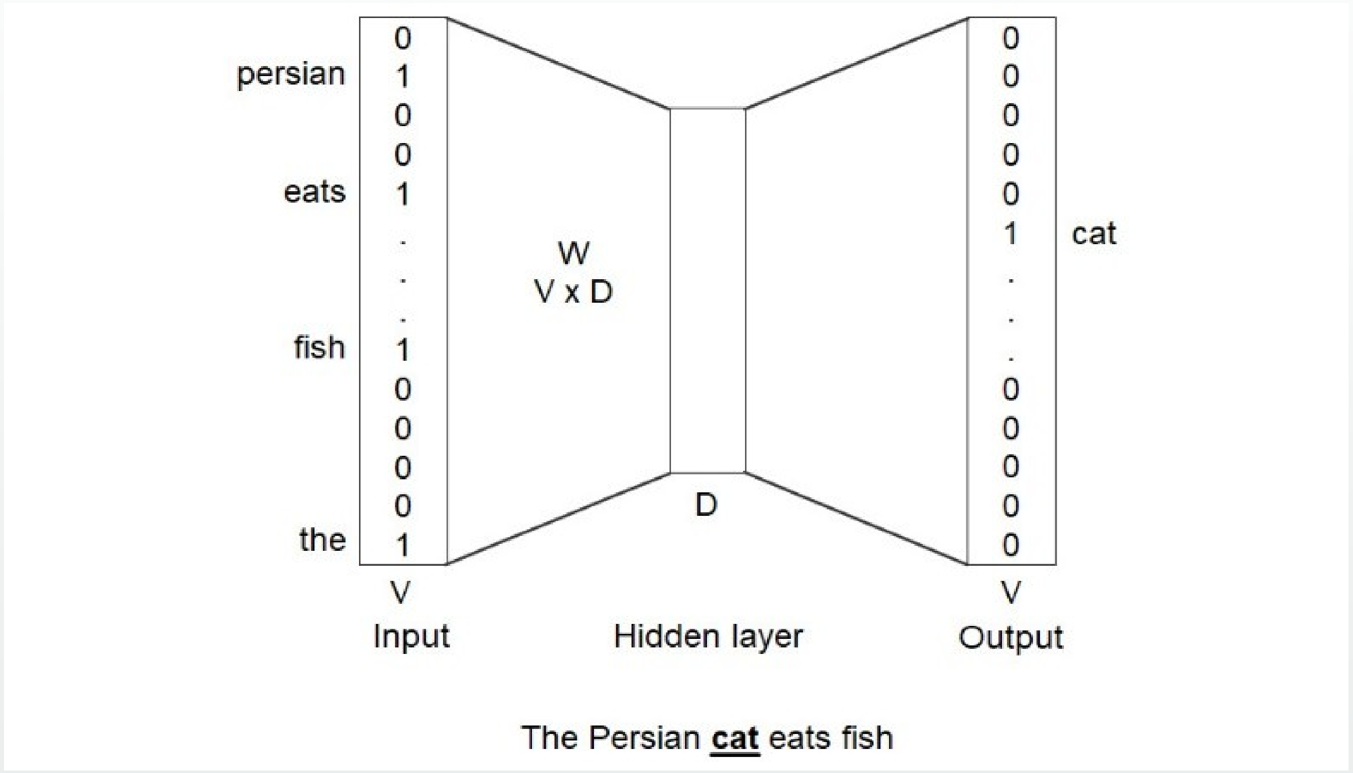




C1	The	persian	cat	eats	fish			
C2		persian	cat	eats	fish	and		
C3			cat	eats	fish	and	hates	
C4				eats	fish	and	hates	bathing

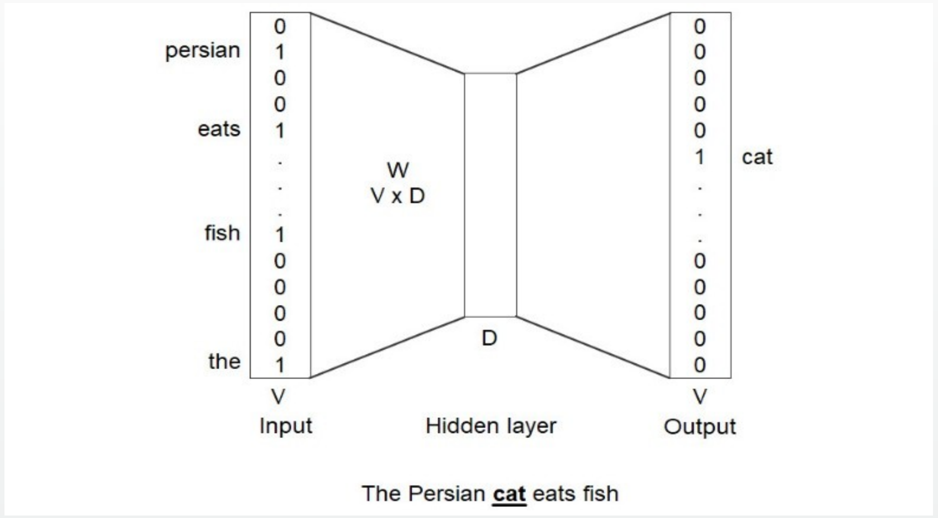
C1, **C2**, **C3**, and **C4** denote the contexts for each window. In **C3**, "fish" is the target word, which is predicted using the terms "cat", "eats", "and", and "hates".

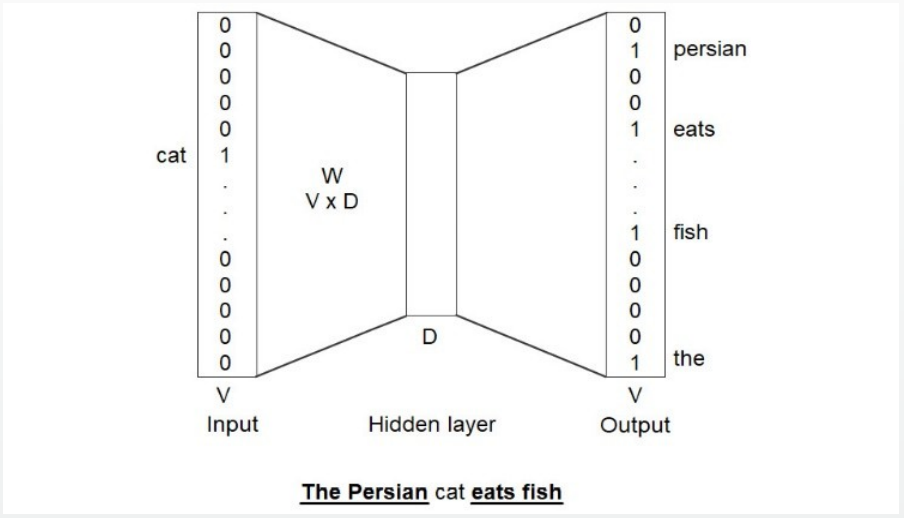
Word2vec: Training



The output layer is for the target term and is one-hot encoded with *V* outputs, one for each term – the predicted term, which is `'cat'`. The input layer for the context terms is also size *V*. The hidden layer is of dimensionality *V x D* (where *D* is the dimension of the vectors). This D-length vector for a term is our word embedding for that term.

CBOW (Left) versus Skip-gram (Right)

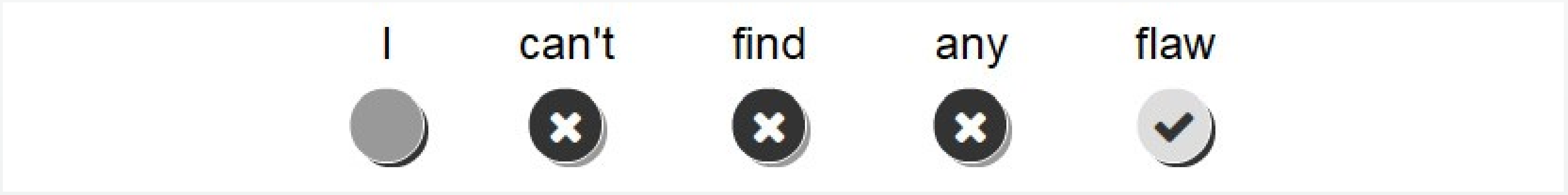




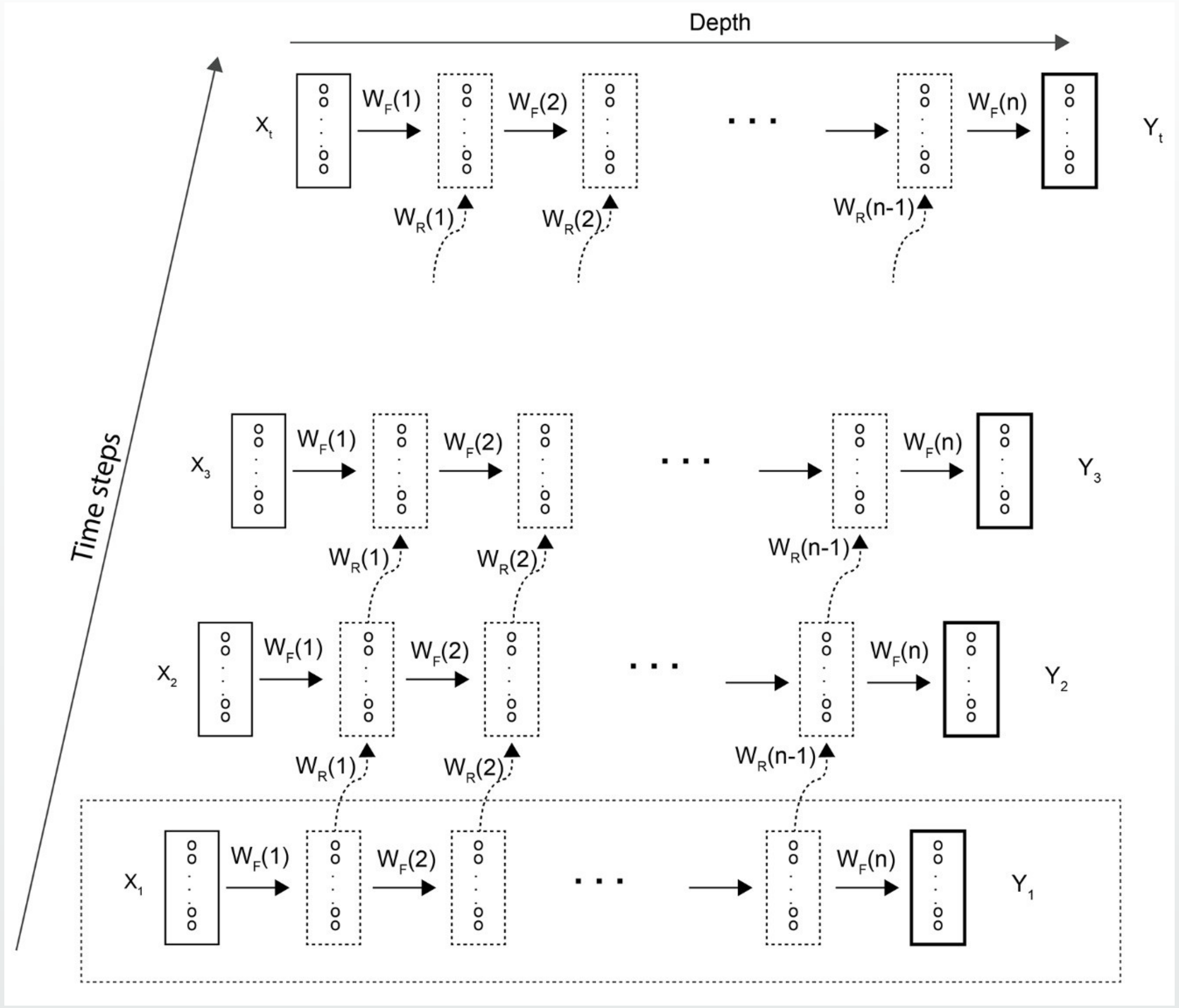
The Skip-gram approach predicts the context words based on the central target word. This flips the formulation of CBOW, where the context words are used to predict the target word.

When you have a small dataset, the smoothing that's done by CBOW is desirable. If you have a small/moderately sized dataset, and if you are concerned about the representation of **rare terms**, then Skip-gram is a good option.

Recurrent Neural Network



Recurrent Neural Network



Long-Range Dependence/Influence

Fill in the blank with a missing country name:

"After a top German university granted her admission for her Masters in Dentistry, Hina was extremely excited to start this new phase of her career with international exposure and couldn't wait till the end of the month to book her flight to ____."

The Vanishing Gradient Problem

We need to be able to handle long-range dependencies. In the context of deep learning models and RNNs, this would mean that learning (or the backpropagation of errors) needs to happen smoothly and effectively over many time steps.

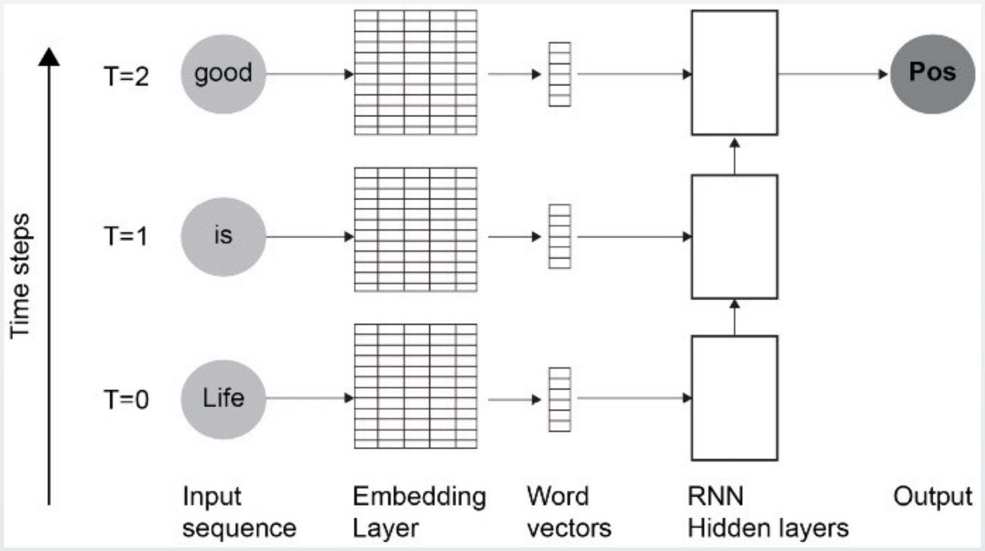
As the model gets more and more layers, backpropagating the errors all the way back to the initial layers becomes increasingly difficult. Layers close to the output will be "learning"/updated at a good pace, but by the time the error propagates to the initial layers, its value will have diminished greatly and have little or no effect on the parameters for the initial layers.

The Embedding Layer

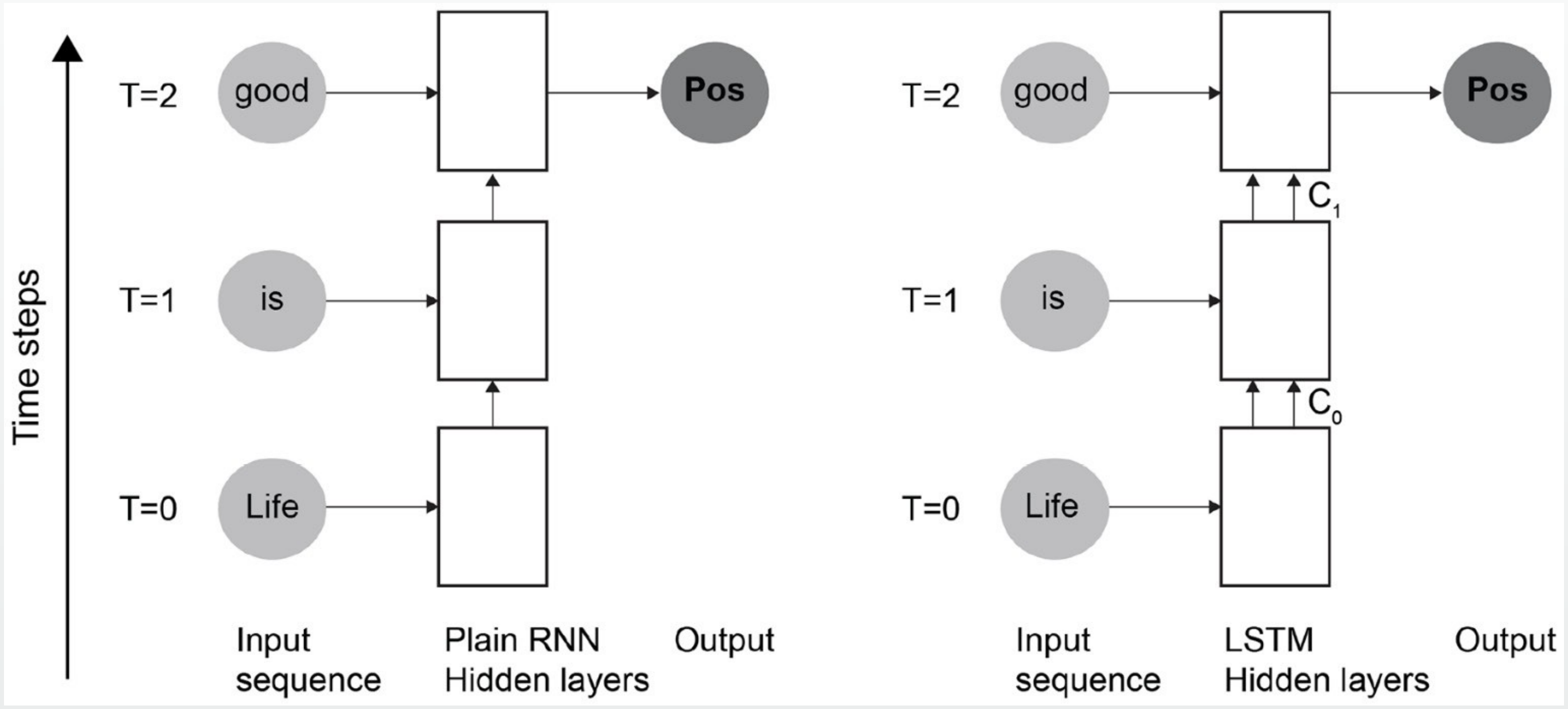
The functionality of the embedding layer is two-fold:

- For any input term, perform a lookup and return its word embedding/vector
- During training, learn these word embeddings

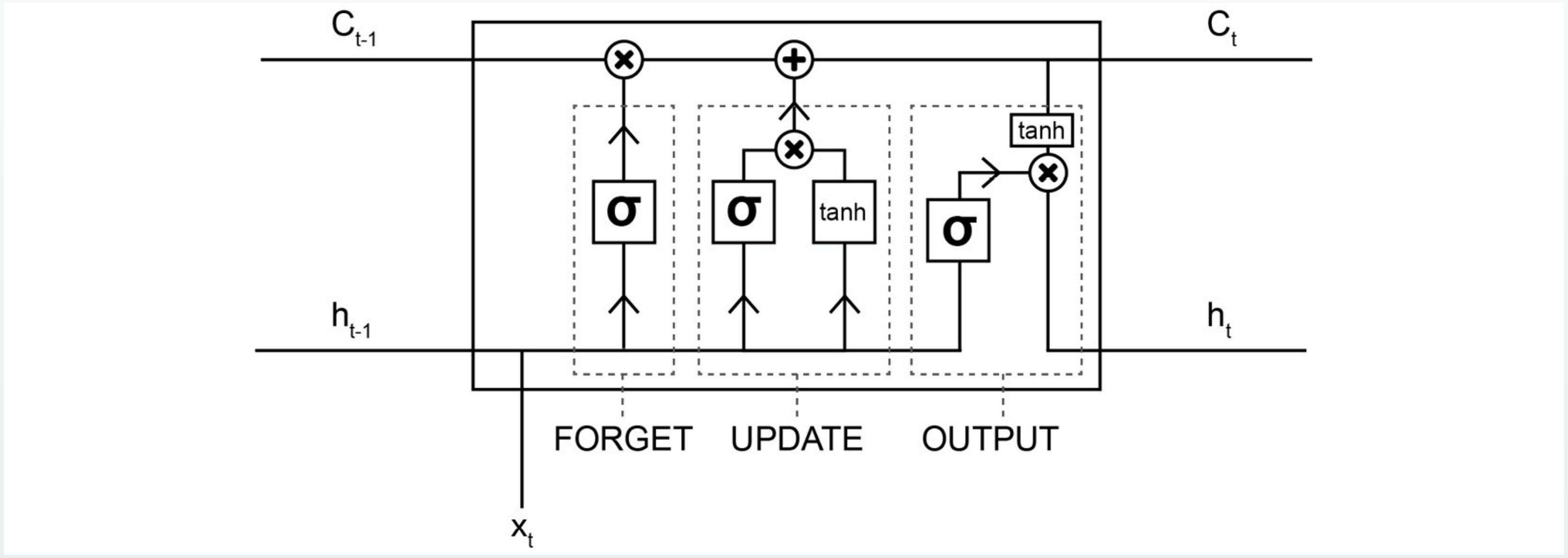
an input sentence "**Life is good**", with the term "**Life**" coming in at time step **T=0** and "**good**" appearing at time step **T=2**. The model will process the inputs one by one, using the embedding layer to look up the word embeddings that will be passed to the hidden layers. The classification will be done when the final term, "**good**", is processed at time step **T=2**. We'll use Keras to build and train our models:



LSTM versus Plan RNN



LSTM Gate



C_{t-1}
C_t
tanh
σ
σ
tanh
σ
h_{t-1}
h_t
FORGET
UPDATE
OUTPUT
x_t