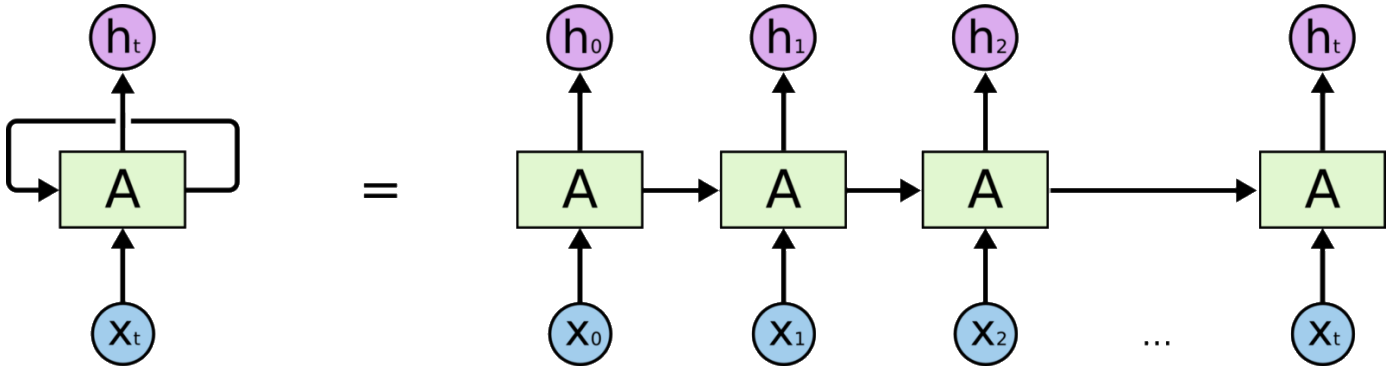
بنام خداوند جان و خرد



Recurrent Neural Networks

Mohammad Taher Pilehvar

Deep Learning

https://teias-courses.github.io/dl99

Memory

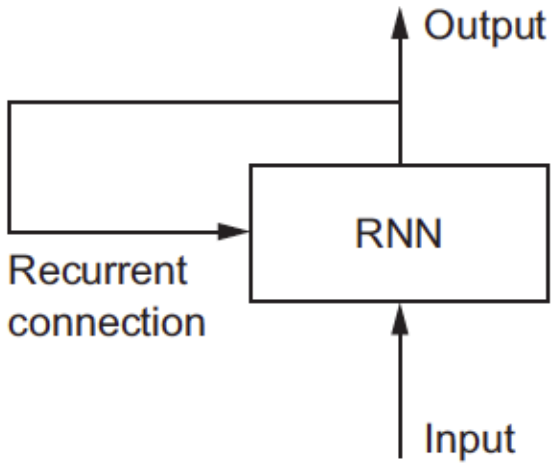
- Dense and ConvNet have no memory!
  - They process the input shown to them independently (with no state in between them)
- These are called *feedforward networks*

- However, we read sentences word by word, keeping a memory of what came before

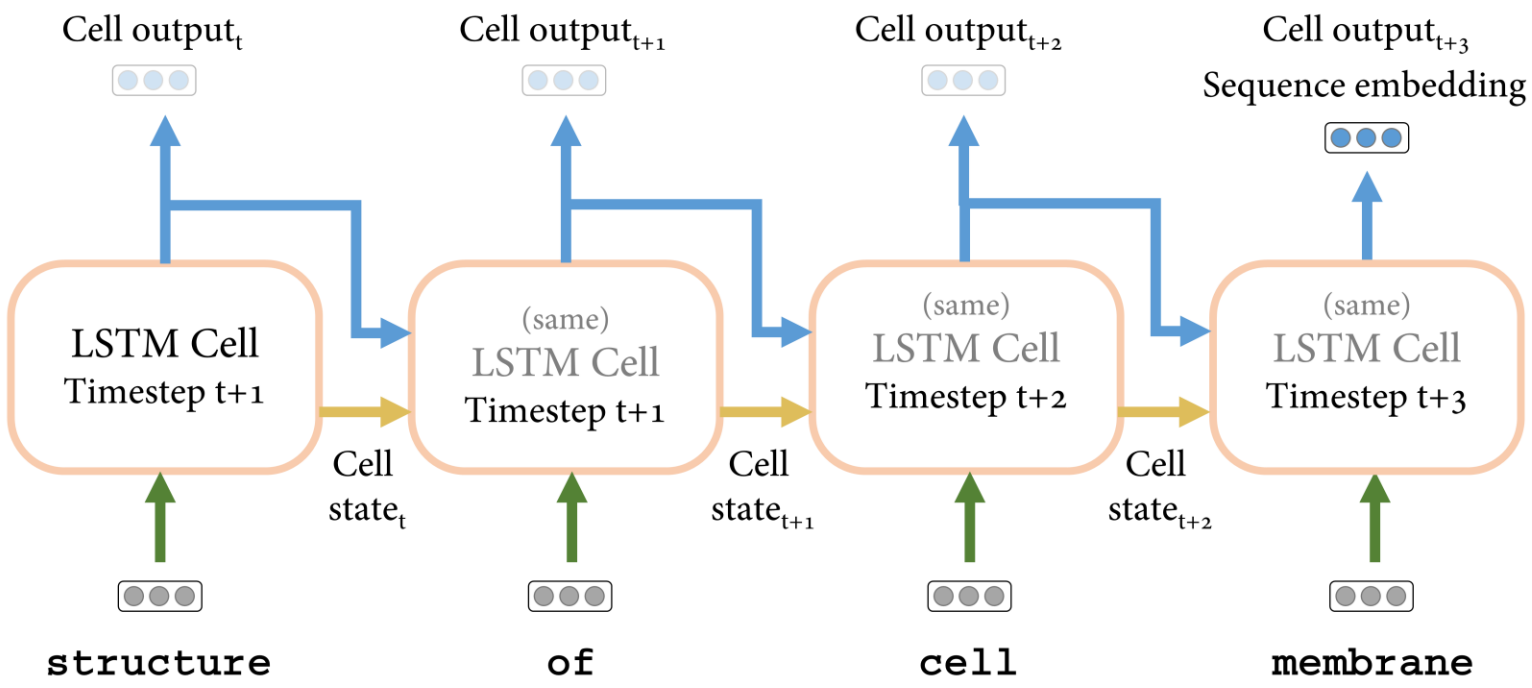
Recurrent Neural Network

- Still, a sequence is a single data point

- What changes is:
  - This data point is no longer processed in a single step; rather, the network internally loops over sequence elements.



Recurrent Neural Networks



In this part

- Deep-learning models that can process text (understood as sequences of word or sequences of characters), timeseries, and sequence data in general.

- ~~Two~~ fundamental deep-learning algorithms for sequence processing:
  - Recurrent neural networks
  - 1D convnets
  - (new) Transformers

Applications of sequence processing

- Document classification and timeseries classification
  - Identifying the topic of an article or the author of a book
- Timeseries comparisons
  - Estimating how closely related two documents or two stock tickers are
- Sequence-to-sequence learning
  - Decoding an English sentence into French
- Sentiment analysis
  - classifying the sentiment of tweets or movie reviews as positive or negative
- Timeseries forecasting
  - predicting the future weather at a certain location, given recent weather data

Text processing

- One of the most widespread forms of sequence data

- Deep learning for natural-language processing is pattern recognition applied to words, sentences, and paragraphs, in much the same way that computer vision is pattern recognition applied to pixels.

Text processing

Like all other neural networks, deep-learning models don't take as input raw text: they only work with numeric tensors.

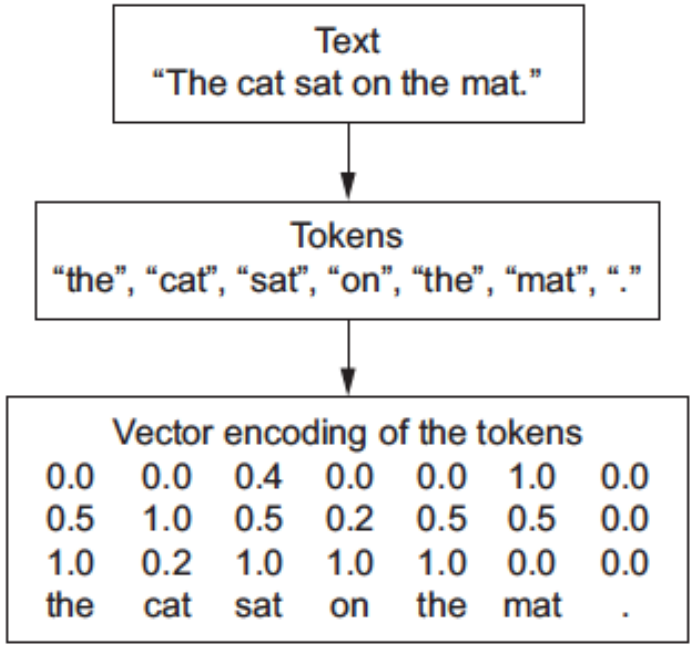
Vectorizing text:

- Segment text into words, and transform each word into a vector.
- Segment text into characters, and transform each character into a vector.
- Extract n-grams of words or characters, and transform each n-gram into a vector. N-grams are overlapping groups of multiple consecutive words or characters.

Tokenization

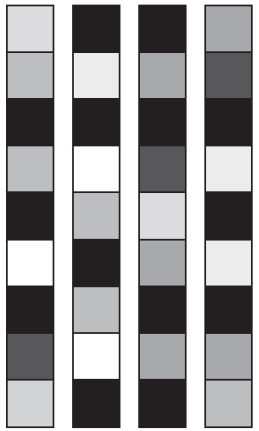
The different units into which you can break down text (words, characters, or n-grams) are called tokens, and breaking text into such tokens is called tokenization.

- All text-vectorization processes consist of applying some tokenization scheme and then associating numeric vectors with the generated tokens.



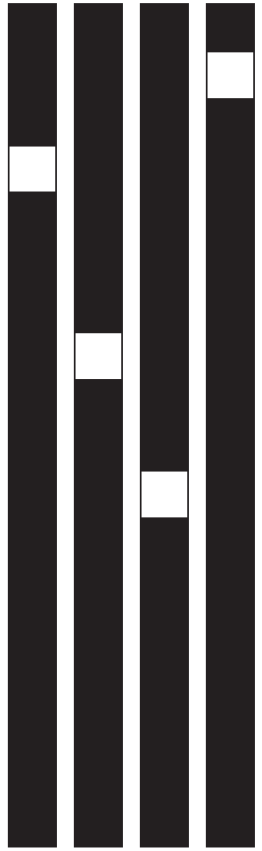
One-hot encoding vs. embeddings

- Tensorboard



One-hot encoding

- Most basic way to turn a token into a vector
- Associating a unique integer index with every word and then turning this integer index i into a binary vector of size N (size of vocabulary)
- The vector is all zeros except for the i^{th} entry, which is 1.

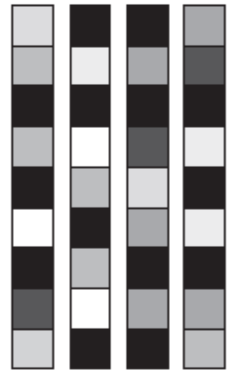


Word embeddings

- Word embeddings pack more information into far fewer dimensions

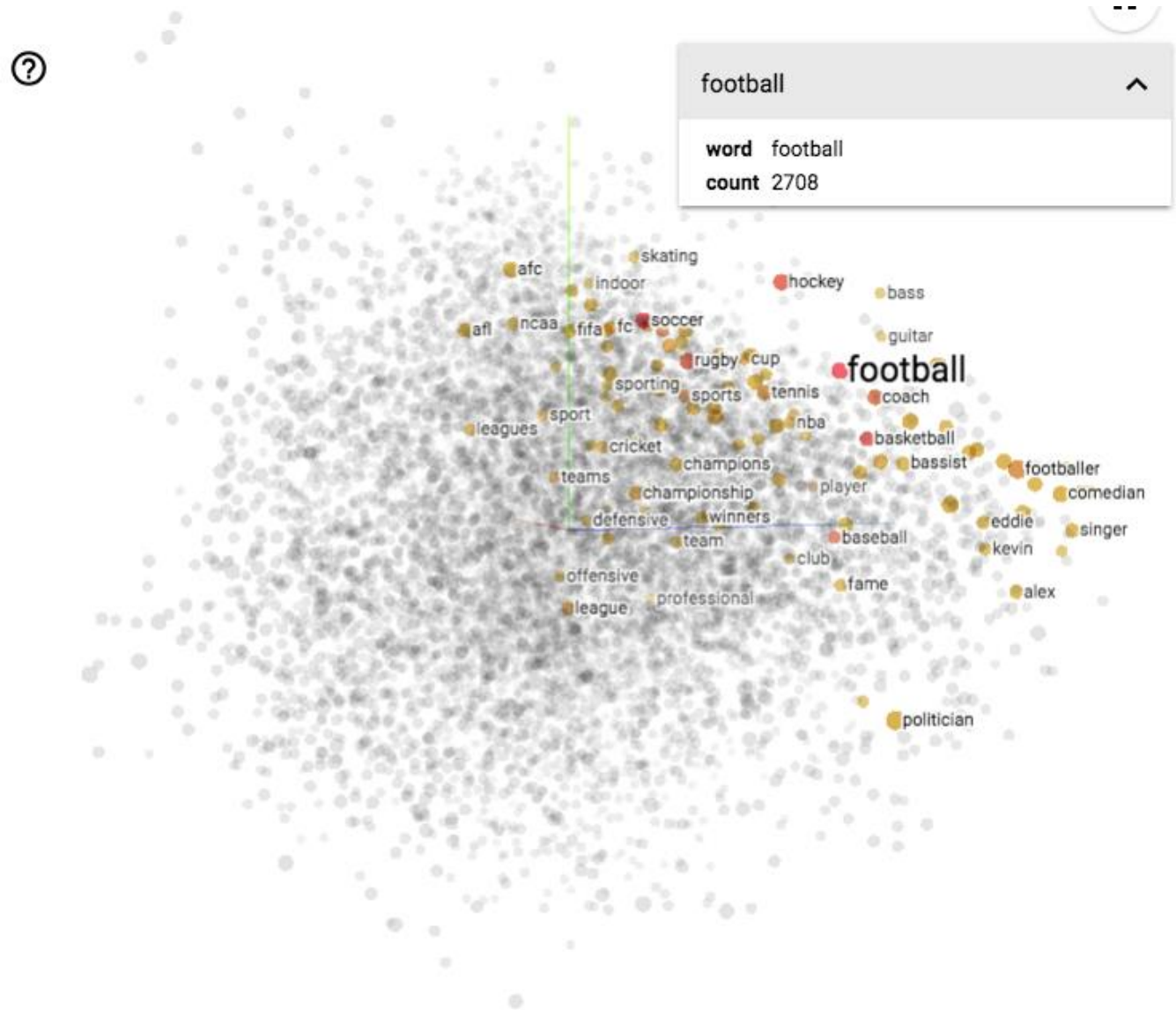
- They can be pre-trained on large amounts of text training data

One-hot word vectors:
- Sparse
- High-dimensional
- Hardcoded



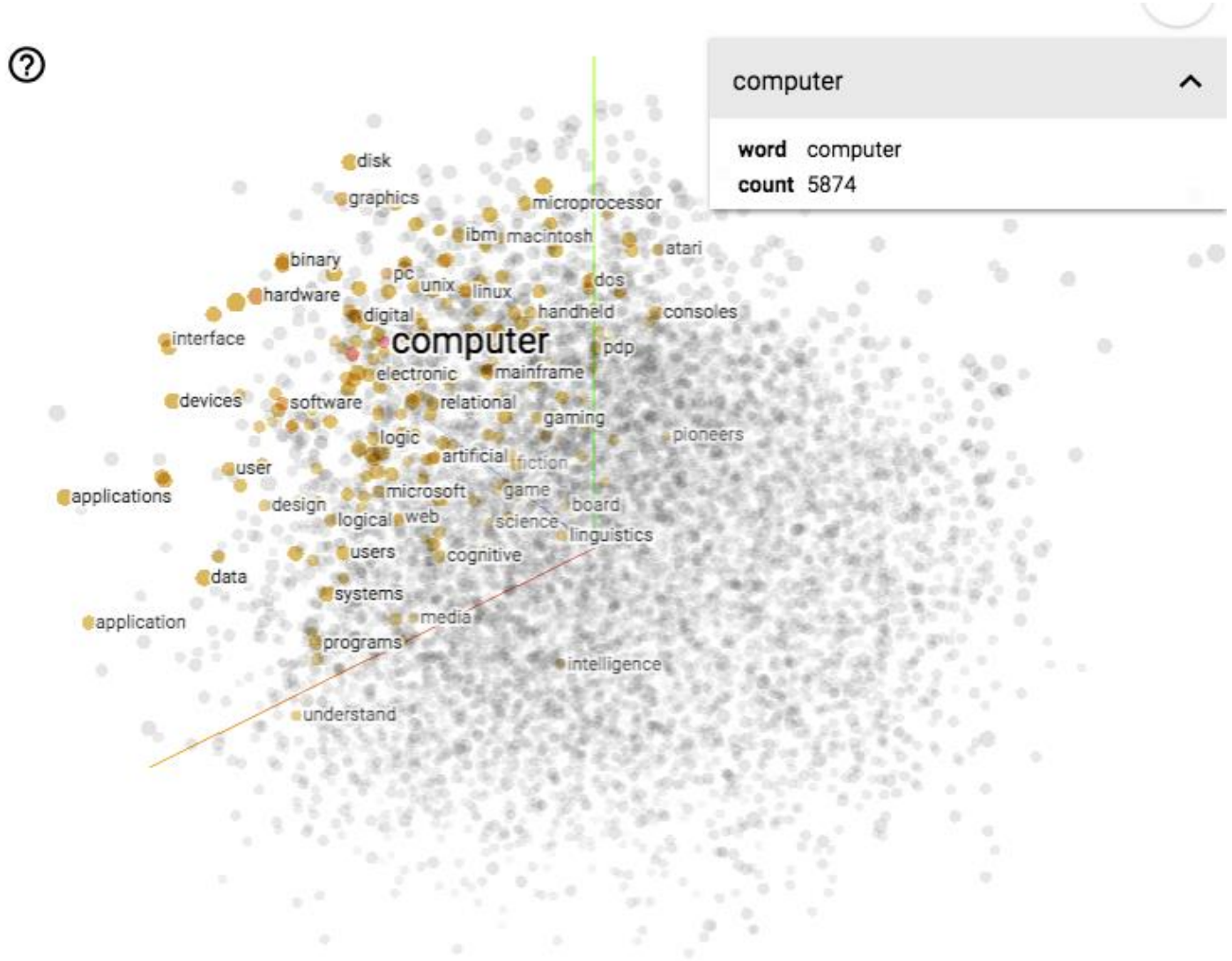
Word embeddings:
- Dense
- Lower-dimensional
- Learned from data

Word embeddings



http://projector.tensorflow.org/

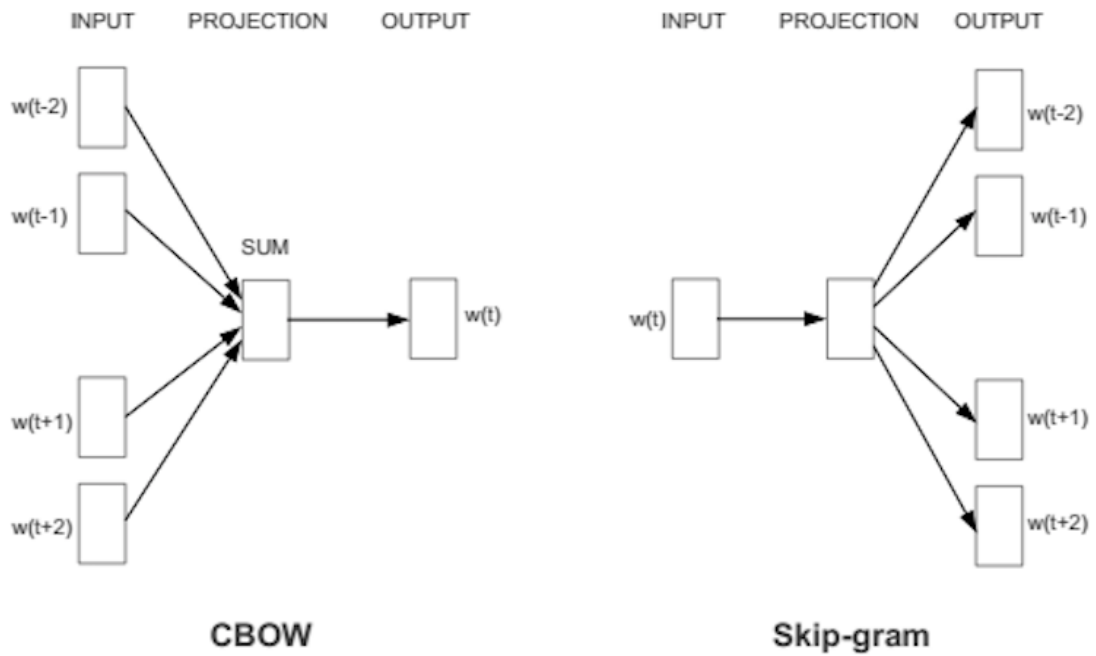
Word embeddings



http://projector.tensorflow.org/

Pre-trained word embeddings

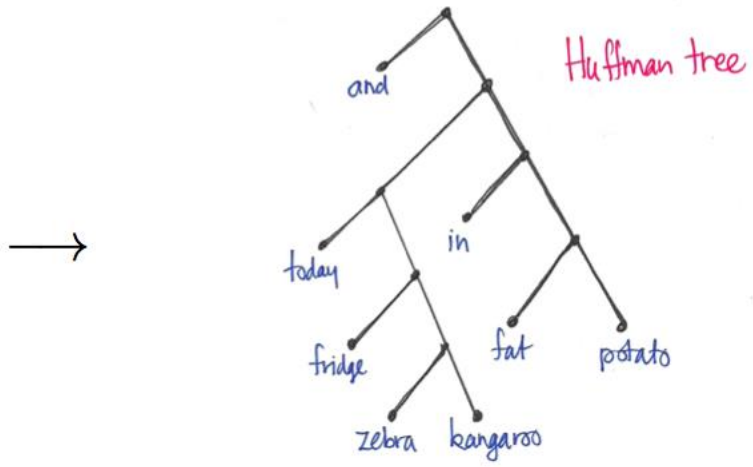
Word2vec



Pre-trained word embeddings

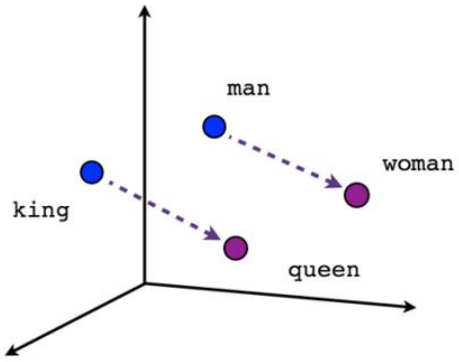
Word2vec: hierarchical softmax

word	count
fat	3
fridge	2
zebra	1
potato	3
and	14
in	7
today	4
kangaroo	2

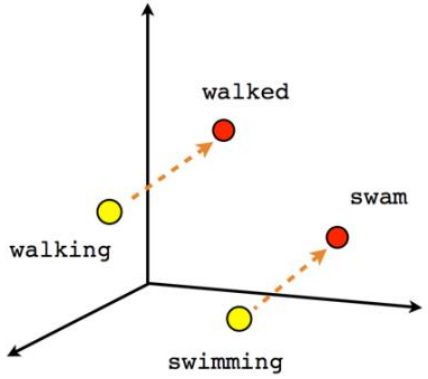


Pre-trained word embeddings

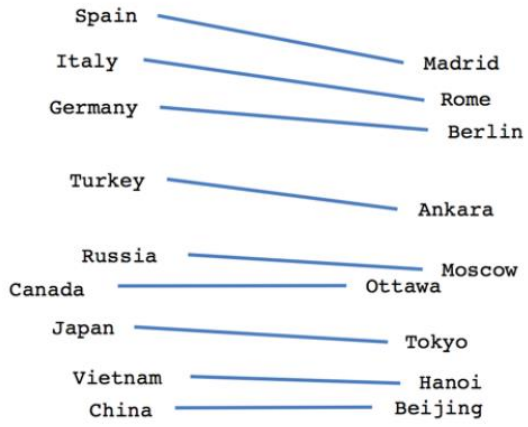
Word2vec properties



Male-Female



Verb tense



Country-Capital

Word embeddings

Two ways to obtain them:

- Learn word embeddings jointly with the main task you care about (such as document classification or sentiment prediction). In this setup, you start with random word vectors and then learn word vectors in the same way you learn the weights of a neural network.

- Load into your model word embeddings that were precomputed using a different machine-learning task than the one you're trying to solve. These are called pretrained word embeddings.

Learning word embeddings

Associate a random vector to each word

What' the problem?

- The problem with this approach is that the resulting embedding space has no structure: for instance, the words *accurate* and *exact* may end up with completely different embeddings, even though they're interchangeable in most sentences.

Learning word embeddings

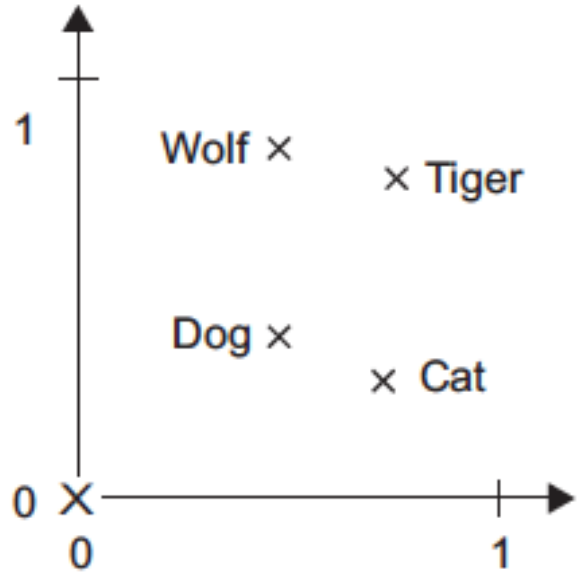
The geometric relationships between word vectors should reflect the semantic relationships between these words.

- In a reasonable embedding space, you would expect synonyms to be embedded into similar word vectors; and in general, you would expect the geometric distance between any two word vectors to relate to the semantic distance between the associated words

Learning word embeddings

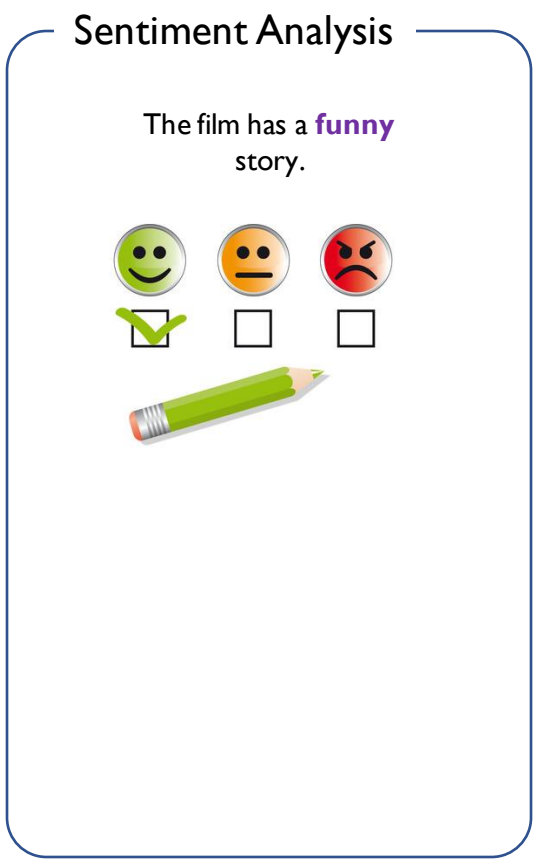
- In addition to distance, you may want specific directions in the embedding space to be meaningful.

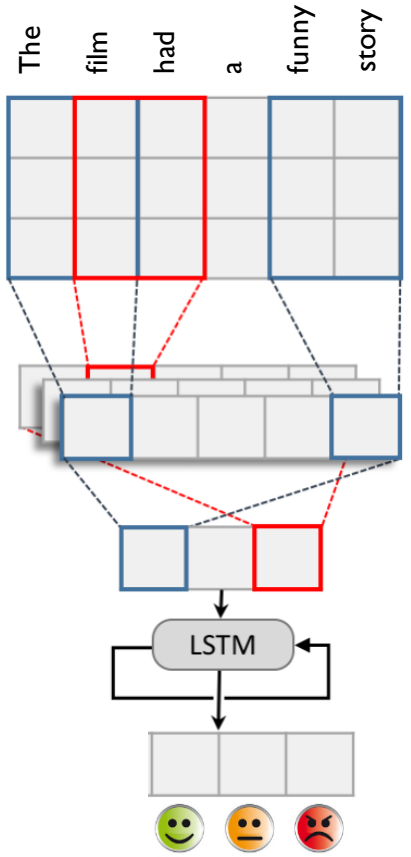
- In real-world word-embedding spaces, common examples of meaningful geometric transformations are "gender" vectors and "plural" vectors.
  - female + king = queen
  - plural + king = kings



VSM: Generalisation Power

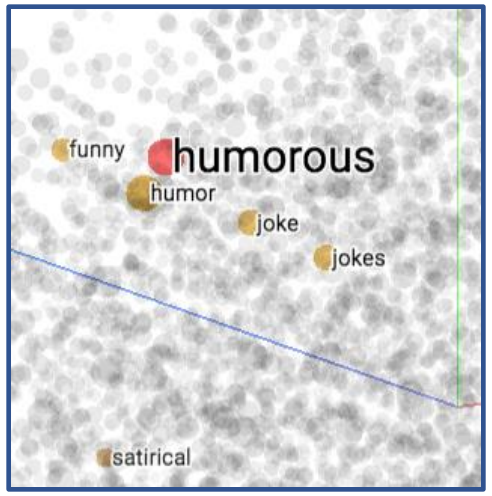
Vector representations can increase generalisation power of NLP systems

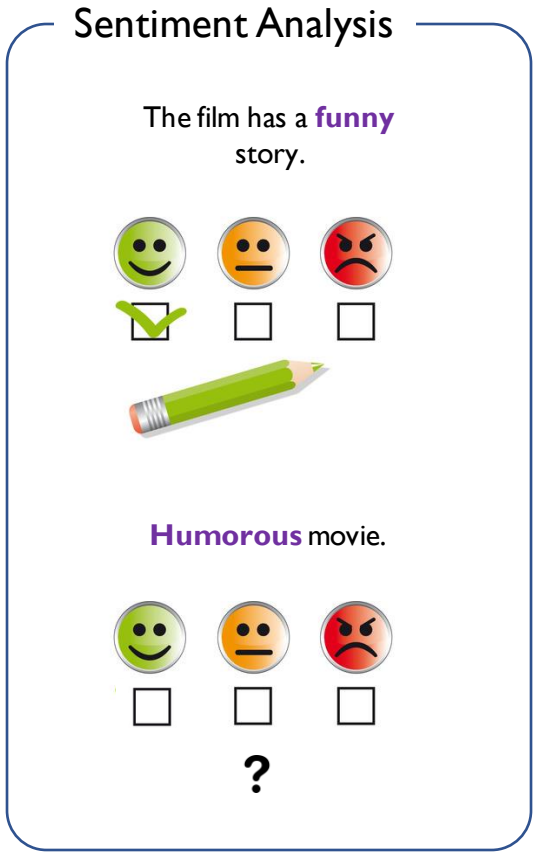


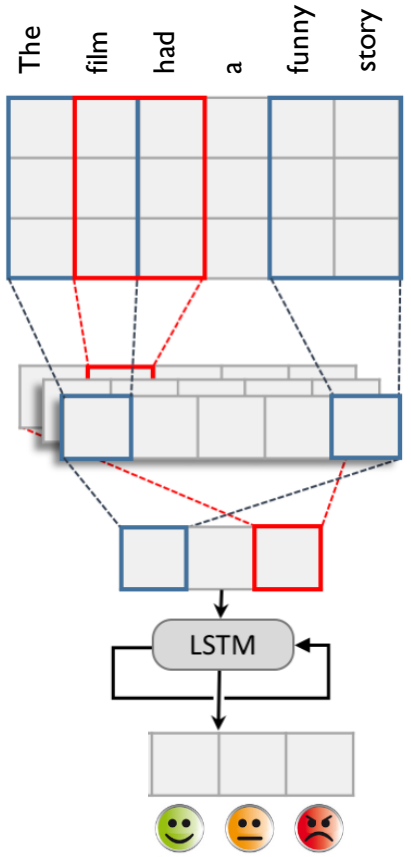


VSM: Generalisation Power

Vector representations can increase generalisation power of NLP systems

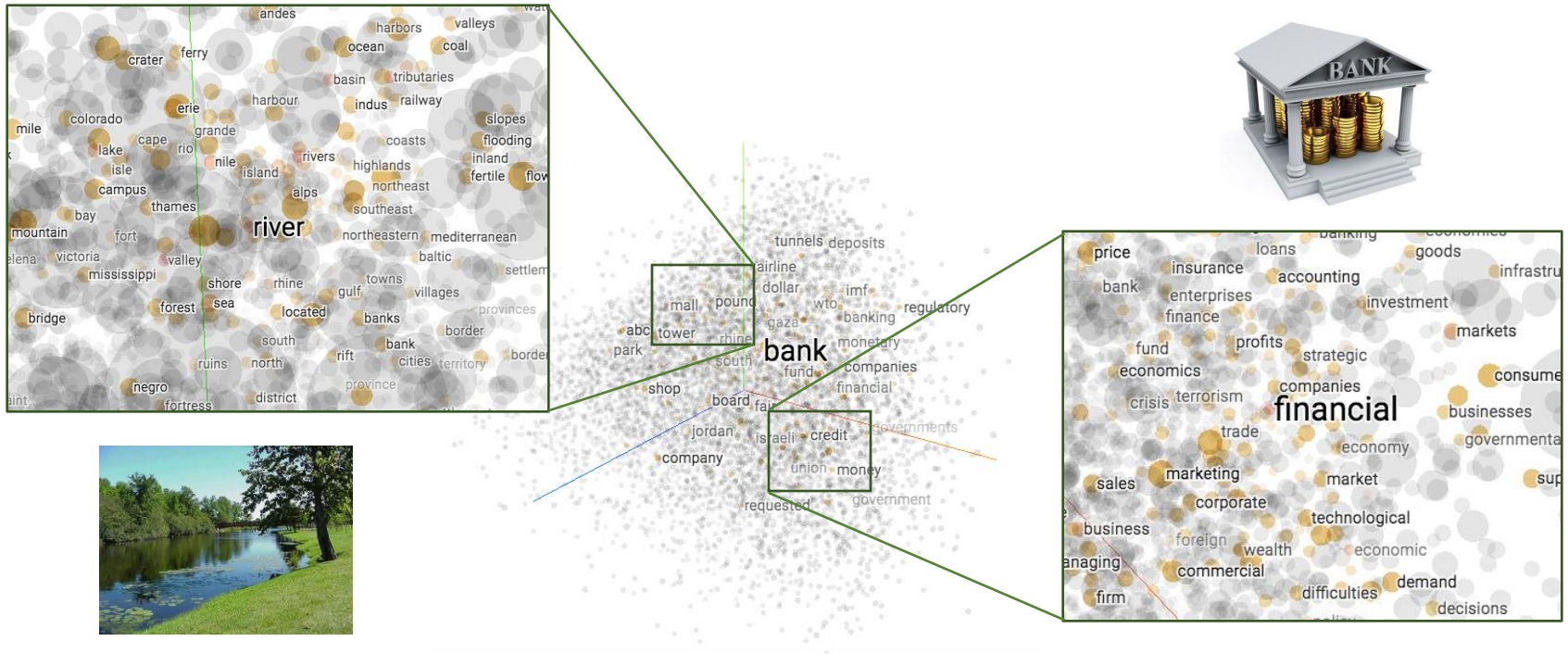






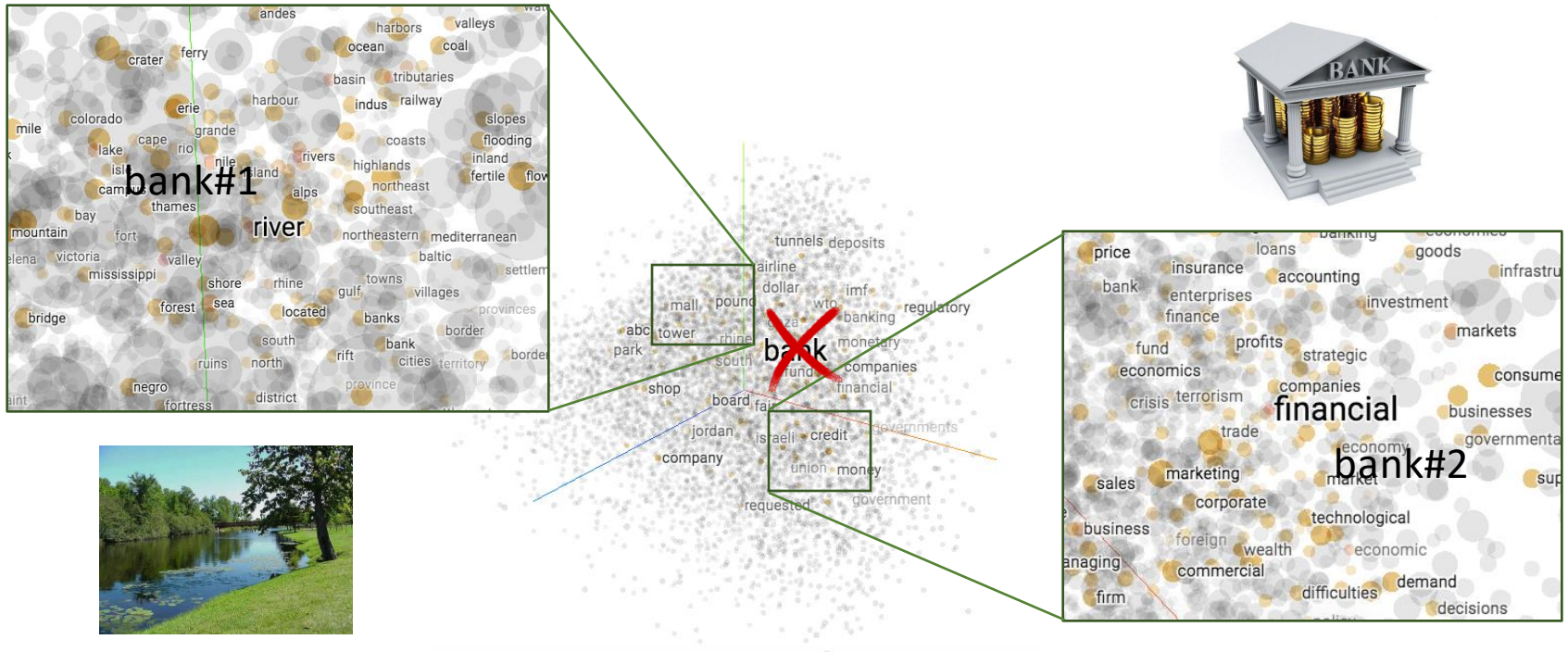
Meaning Conflation Deficiency

Word representations **conflate different meanings** of a word into a single representation: they cannot capture **polysemy**



Word Sense Representations

One can address this deficiency by representing **individual meanings of words (word senses)**



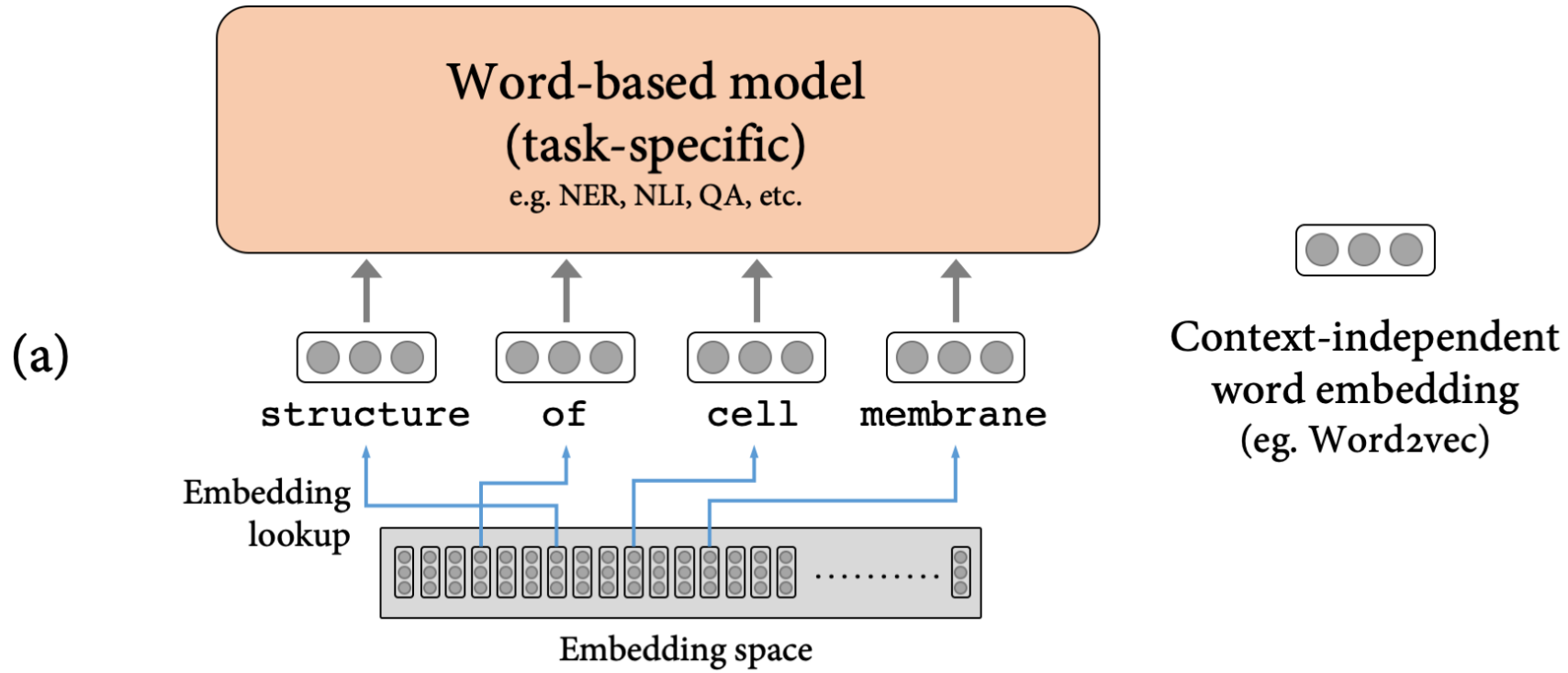
Learning word embeddings

What makes a good word-embedding space depends heavily on your task:

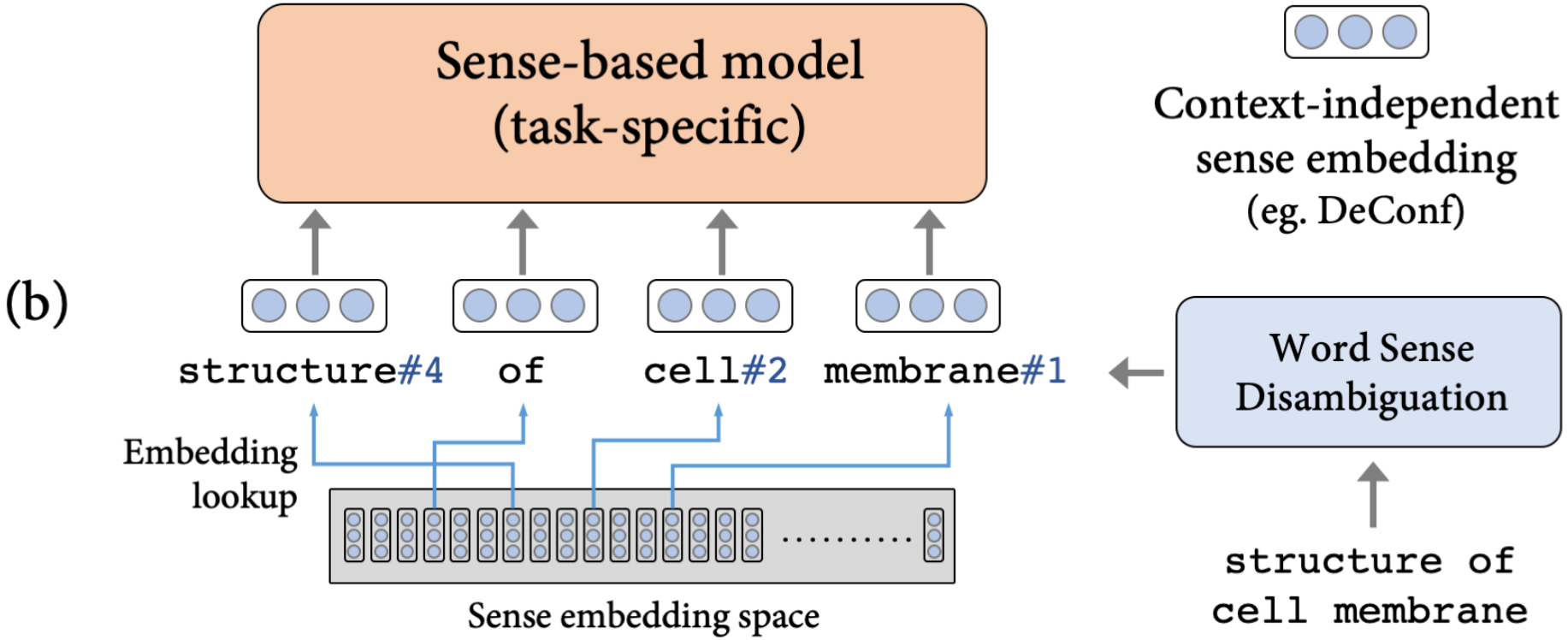
> The perfect word-embedding space for an English-language movie-review sentiment analysis model may look different from the perfect embedding space for an English language legal-document-classification model, because the importance of certain semantic relationships varies from task to task.

Reasonable to learn a new embedding space with every new task

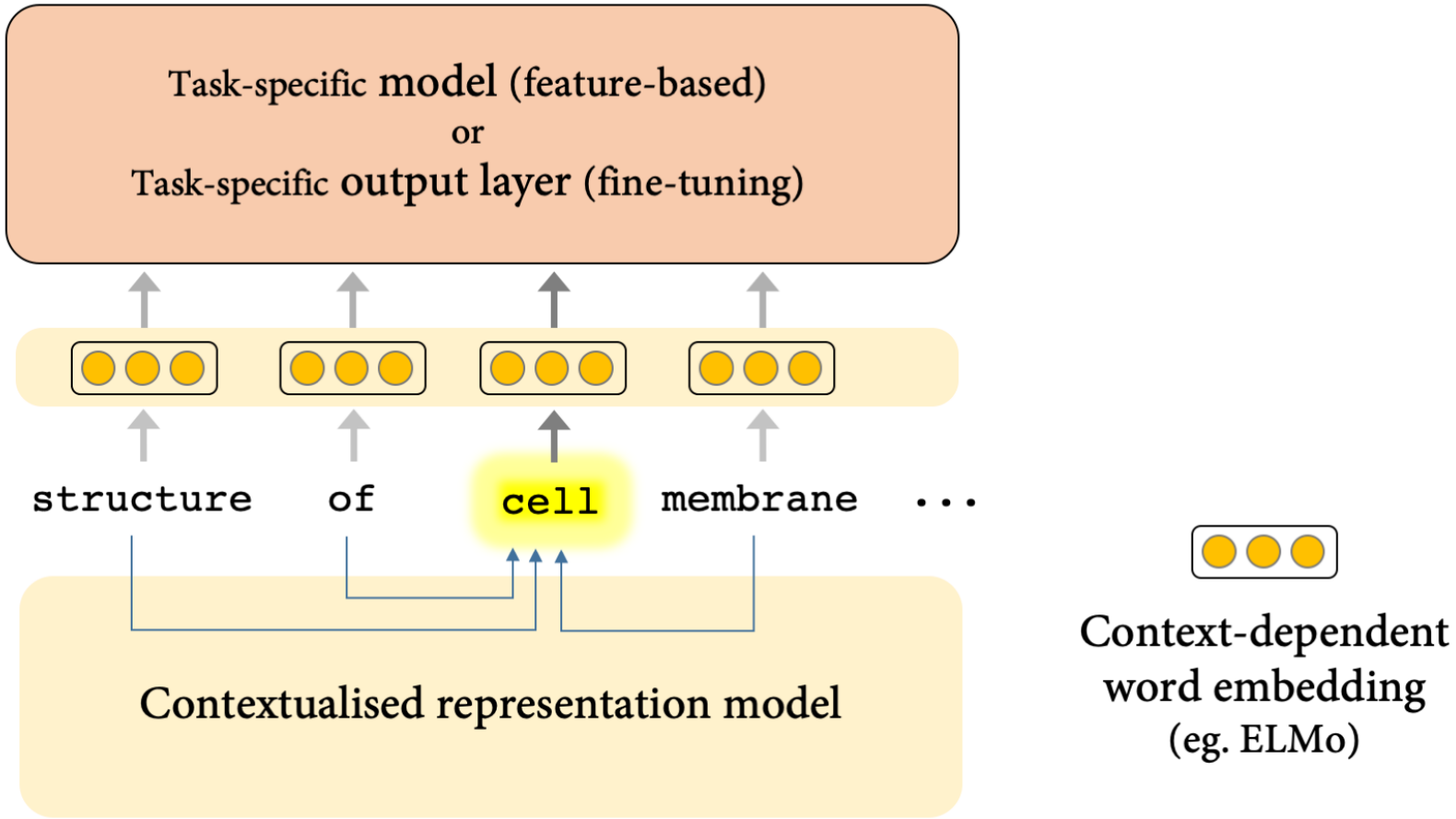
Contextualized representations



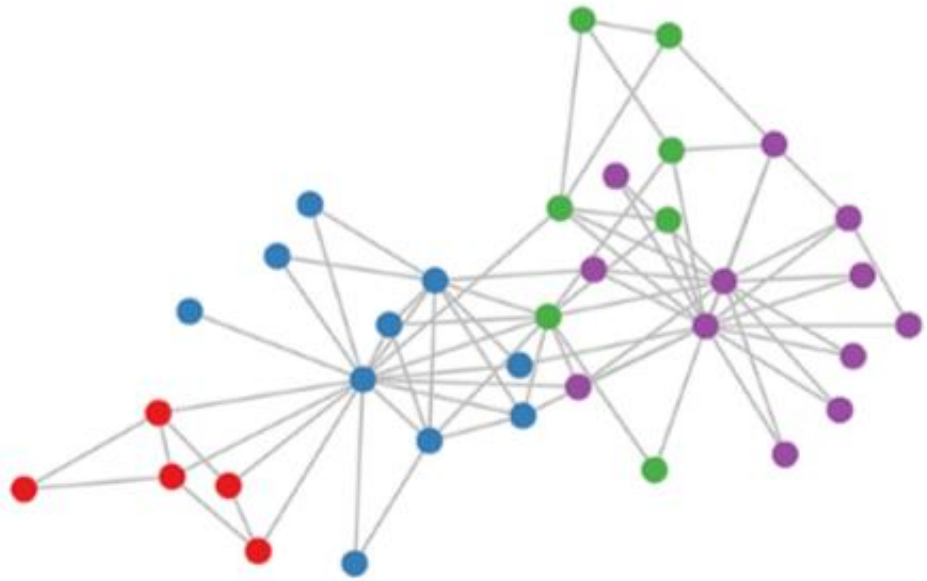
Contextualized representations



Contextualized representations



(Graph embeddings)



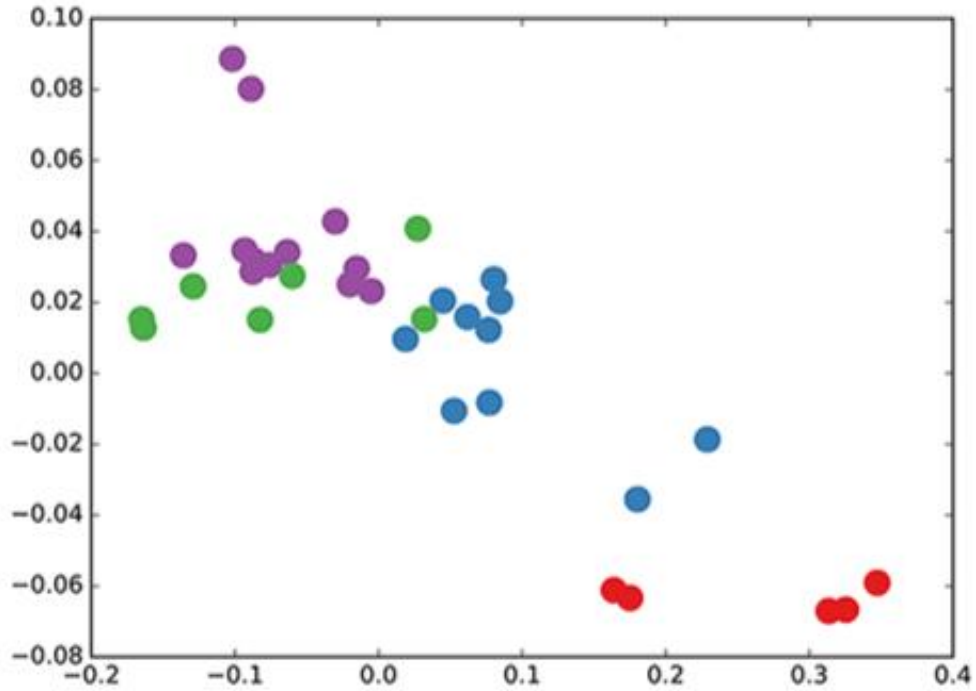
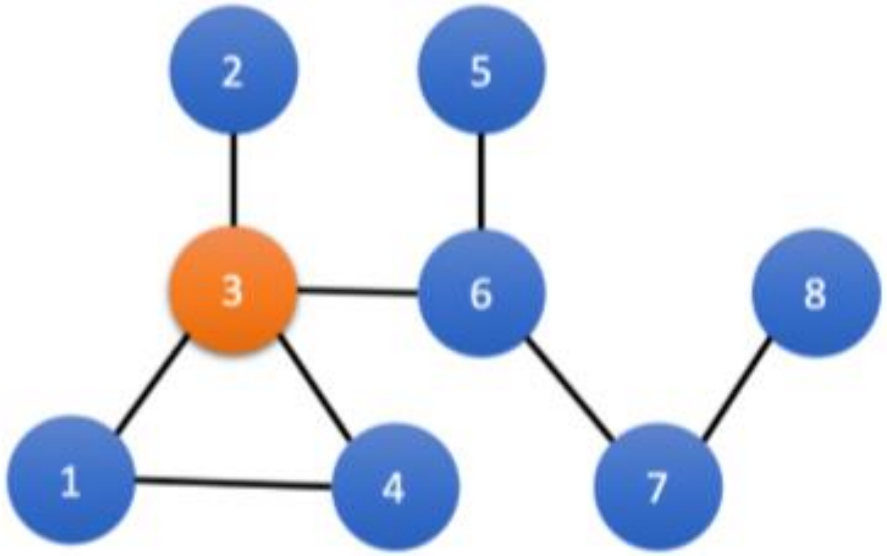


Figure from Kipf and Welling [2017]

(Graph embeddings)



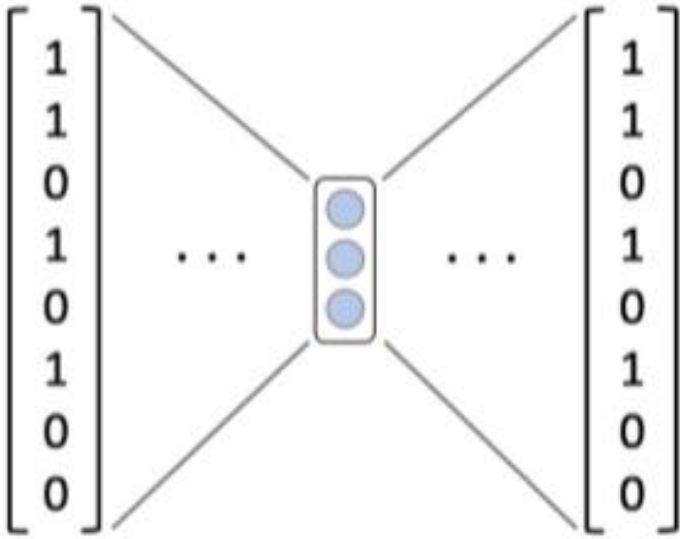
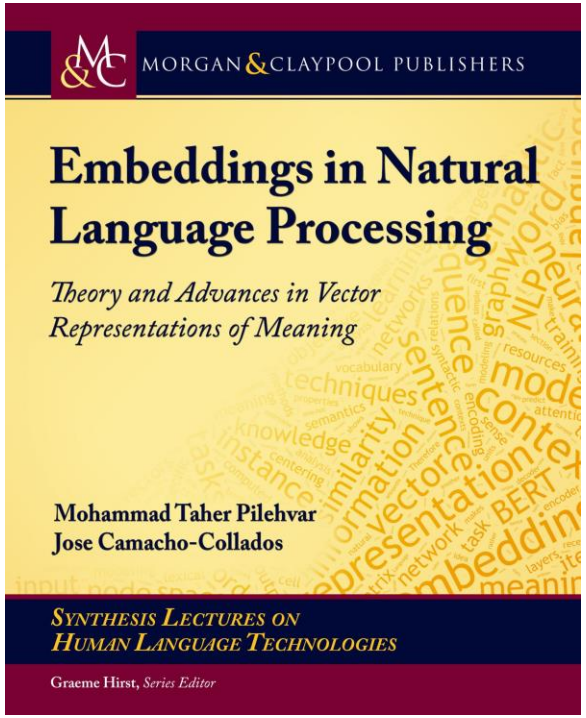


Figure from Kipf and Welling [2017]

Embeddings in NLP

- Read more here
- Book (draft)



Pre-trained word embeddings

Similar in concept to pre-trained ConvNets

> You don't have enough data available to learn truly powerful features on your own, but you expect the features that you need to be fairly generic

- Usually computed using word-occurrence statistics using a variety of techniques, some involving neural networks, others not.

Understanding RNNs

- In Dense and ConvNet, each input is processed independently, with no state kept in between inputs:
  - To process a sequence or a temporal series of data points, you have to show the entire sequence to the network at once: turn it into a single data point.

- In contrast, human processes a text word by word, while keeping memories of what came before.
  - Biological intelligence processes information incrementally while maintaining an internal model of what it's processing, built from past information and constantly updated as new information comes in.

Understanding RNNs

- RNN adopts the same principle, albeit in an extremely simplified version:
  - It processes sequences by iterating through the sequence elements and maintaining a state containing information relative to what it has seen so far.

- The state of the RNN is reset between processing two different, independent sequences.
  - We can still consider one sequence a single data point: a single input to the network.

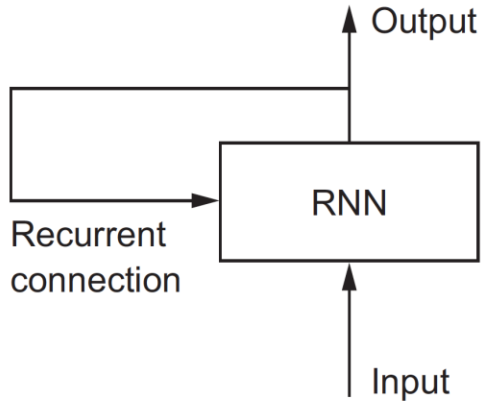


Figure 6.9 A recurrent network: a network with a loop

Understanding RNNs

To make it clear, let's implement the forward pass of a toy RNN in Numpy.

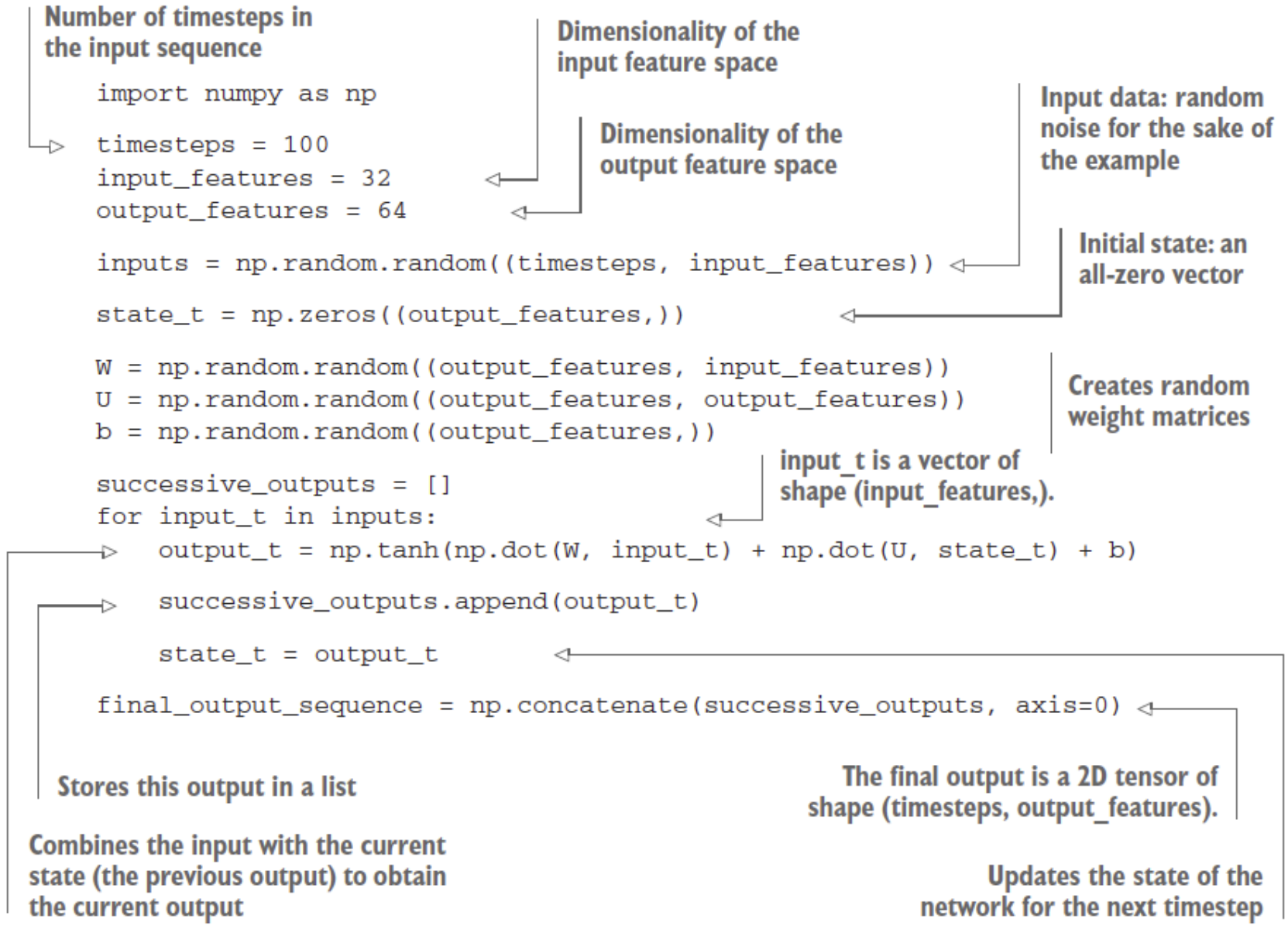
```
state_t = 0                          <-- The state at t
for input_t in input_sequence:       <-- Iterates over sequence elements
    output_t = f(input_t, state_t)
    state_t = output_t               <-- The previous output becomes the state for the next iteration.
```

Understanding RNNs

- You can even flesh out the function `f`: the transformation of the input and state into an output will be parameterized by two matrices, `W` and `U`, and a bias vector.

```
state_t = 0
for input_t in input_sequence:
    output_t = activation(dot(W, input_t) + dot(U, state_t) + b)
    state_t = output_t
```

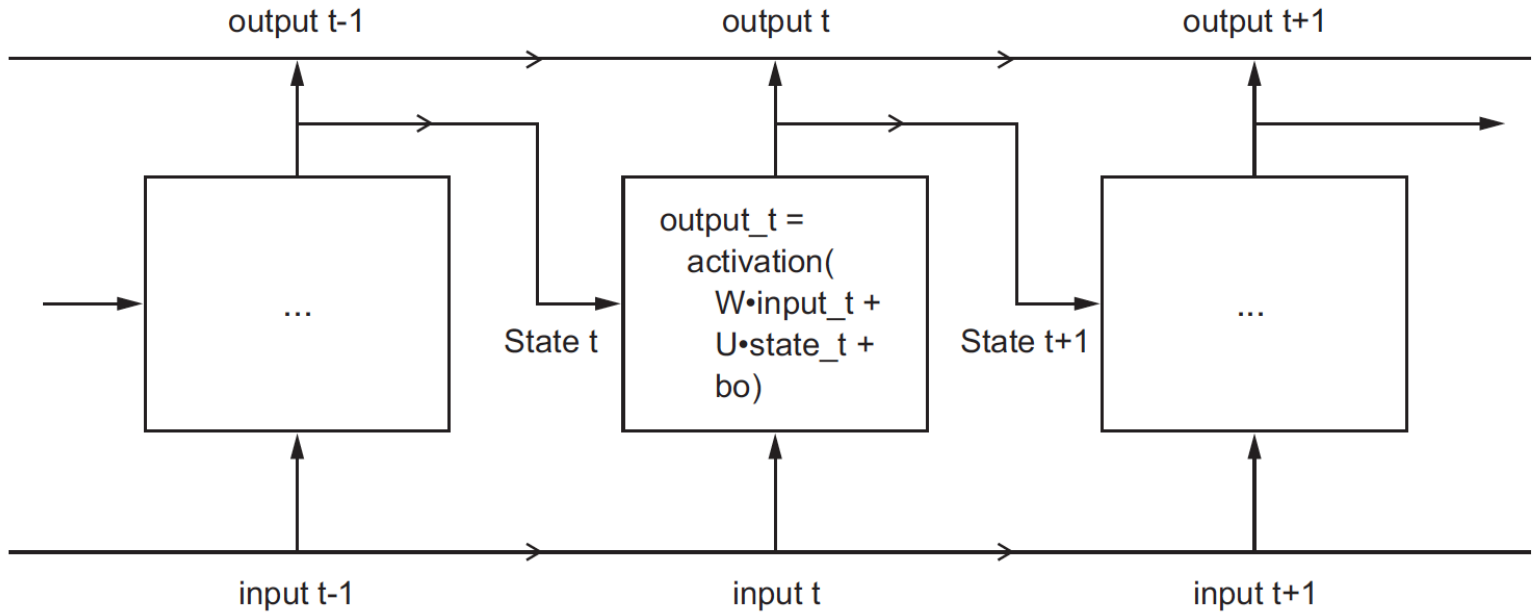
Understanding RNNs



Understanding RNNs

- RNNs are characterized by their step function, such as the following function in this case:

```
output_t = np.tanh(np.dot(W, input_t) + np.dot(U, state_t) + b)
```



Understanding LSTM and GRU layers

SimpleRNN is generally too simplistic to be of real use.

- Although it should theoretically be able to retain at time `t` information about inputs seen many timesteps before, in practice, such long-term dependencies are impossible to learn.

Vanishing gradient problem

Vanishing gradient problem

An effect that is similar to what is observed with non-recurrent networks (feedforward networks) that are many layers deep: as you keep adding layers to a network, the network eventually becomes untrainable.

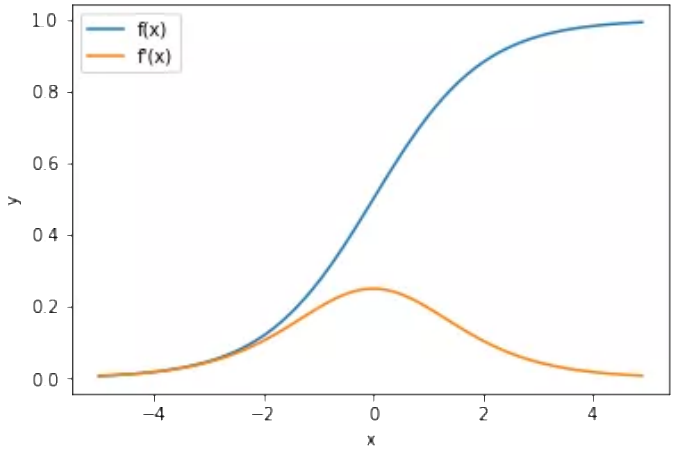
Vanishing gradients

Arises due to the nature of the back-propagation optimization

- The weight and bias values in the various layers within a neural network are updated each optimization iteration by stepping in the direction of the *gradient* of the weight/bias values with respect to the loss function.

Problematic for deep feedforward networks

- Particularly with sigmoid activation functions.
- When the sigmoid function value is either too high or too low, the derivative (orange line) becomes very small i.e. << 1.



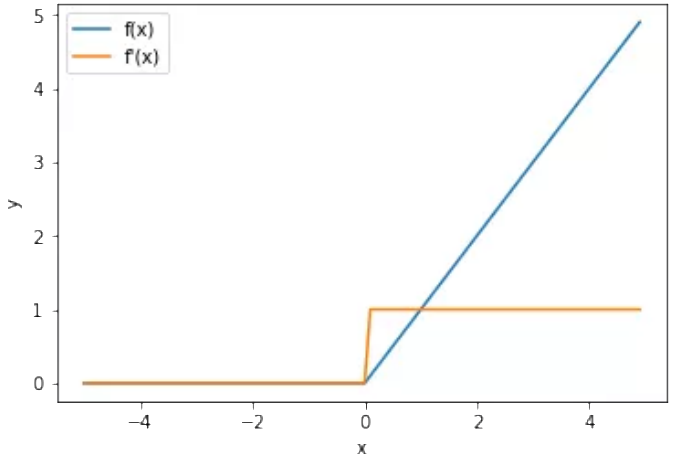
Read more here: https://adventuresinmachinelearning.com/vanishing-gradient-problem-tensorflow/

Vanishing gradients

ReLU activation can partly solve the problem

No degradation of the error signal

But, certain weights can be cancelled out whenever there is a negative input into a given neuron



Vanishing gradients

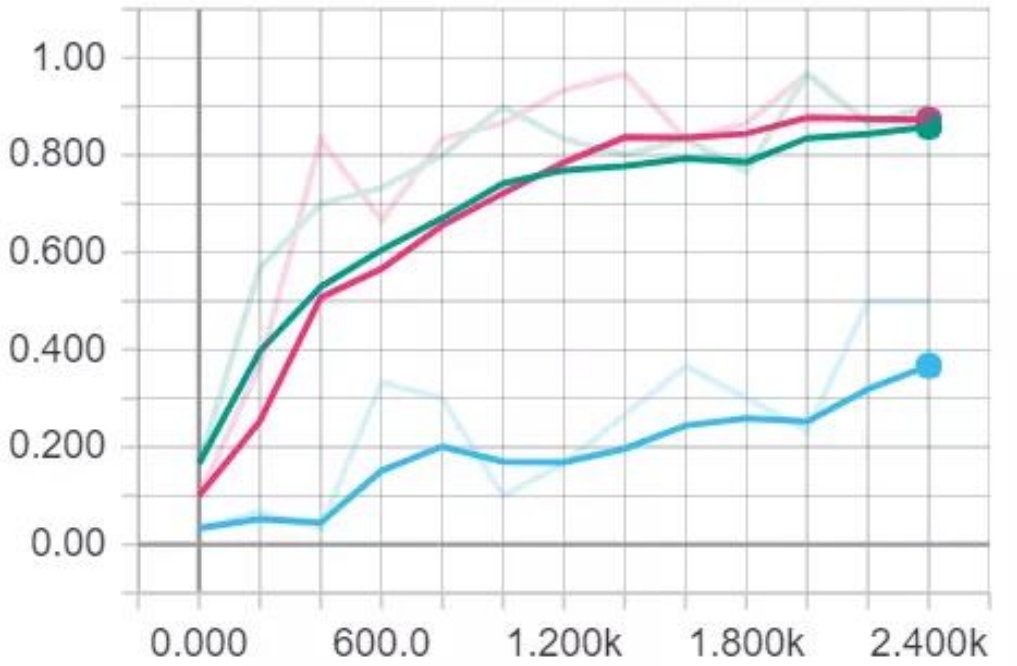
Leaky ReLU

$$f(x) = \max(\alpha x, x)$$

The good thing about the Leaky ReLU activation function is that the derivative when *x* is below zero is alpha – i.e. it is a small but no longer 0.

Vanishing gradients (in practice)

MNIST using a 7-layer densely connected network

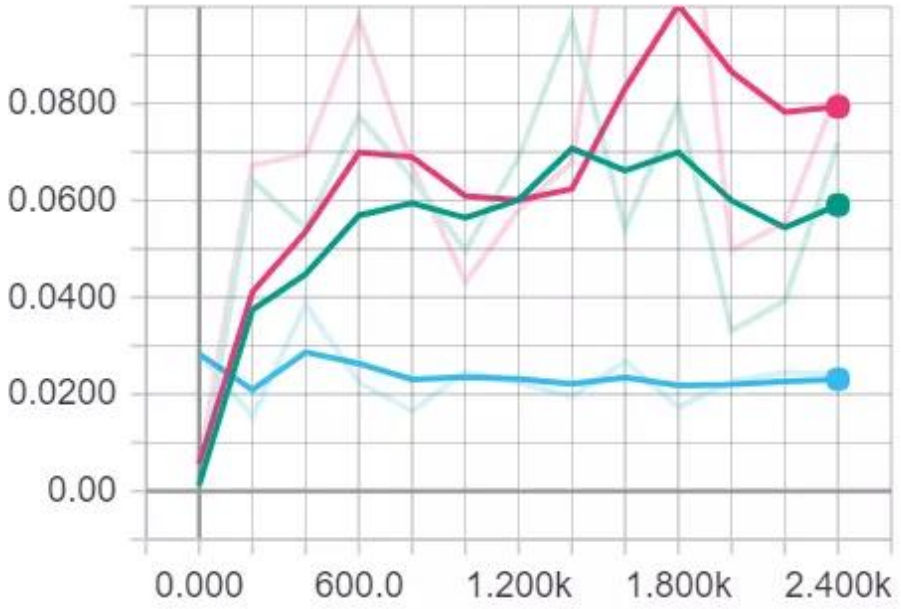


Accuracy of the three activation scenarios
sigmoid (blue),
ReLU (red),
Leaky ReLU (green)

Vanishing gradients (in practice)

MNIST using a 7-layer densely connected network

Mean absolute gradient logs during training

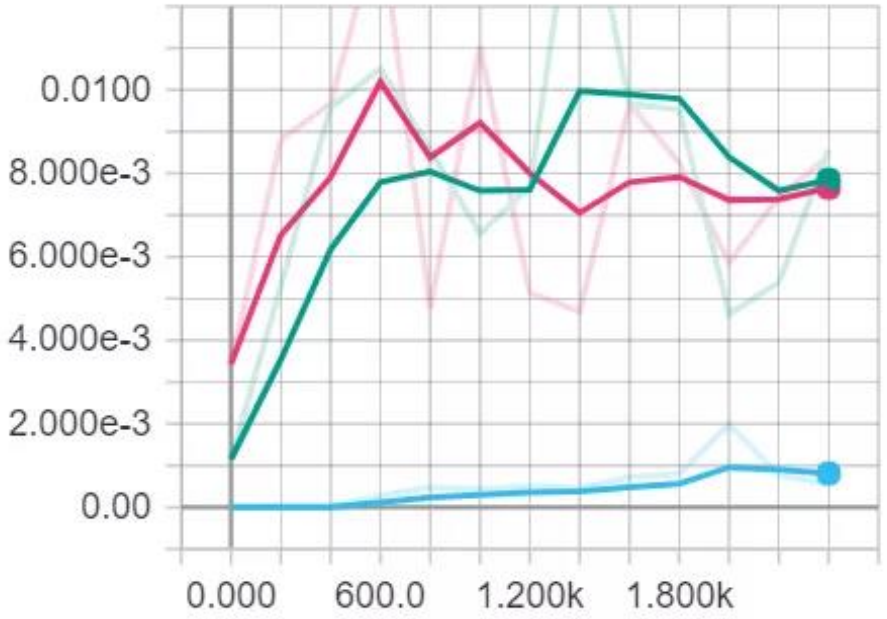


Mean absolute gradients
Output layer (6th layer)
sigmoid (blue),
ReLU (red),
Leaky ReLU (green)

Vanishing gradients (in practice)

MNIST using a 7-layer densely connected network

Mean absolute gradient logs during training



Mean absolute gradients
1st layer
sigmoid (blue),
ReLU (red),
Leaky ReLU (green)

From SimpleRNN to LSTM

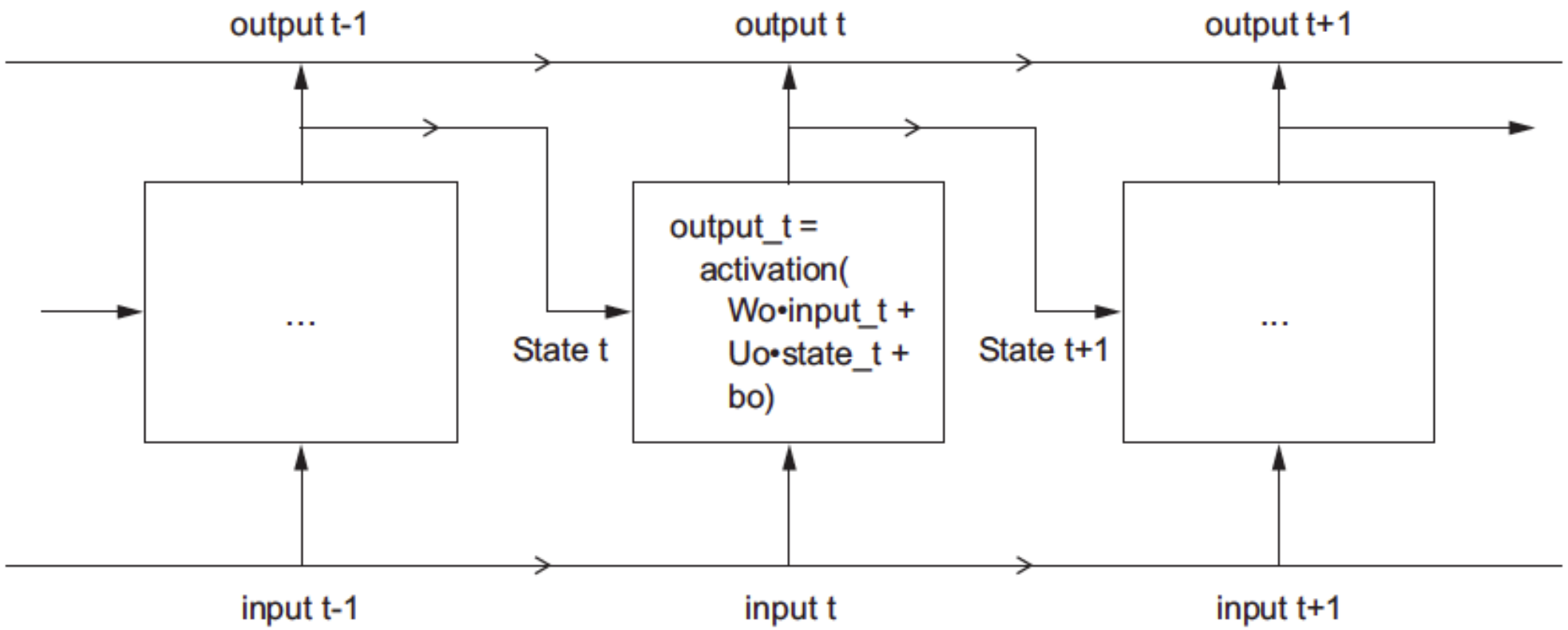


Figure 6.13 The starting point of an `LSTM` layer: a `SimpleRNN`

From SimpleRNN to LSTM

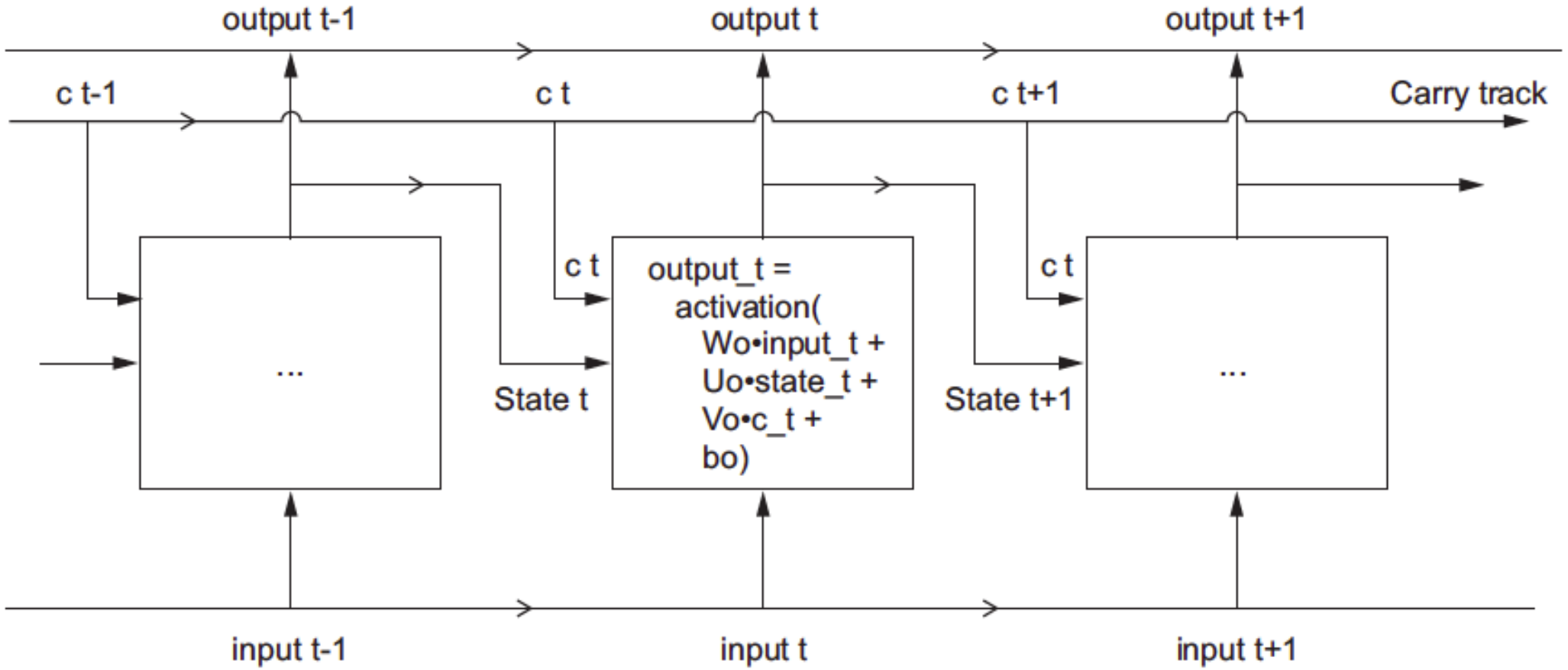


Figure 6.14 Going from a `SimpleRNN` to an `LSTM`: adding a carry track

From SimpleRNN to LSTM

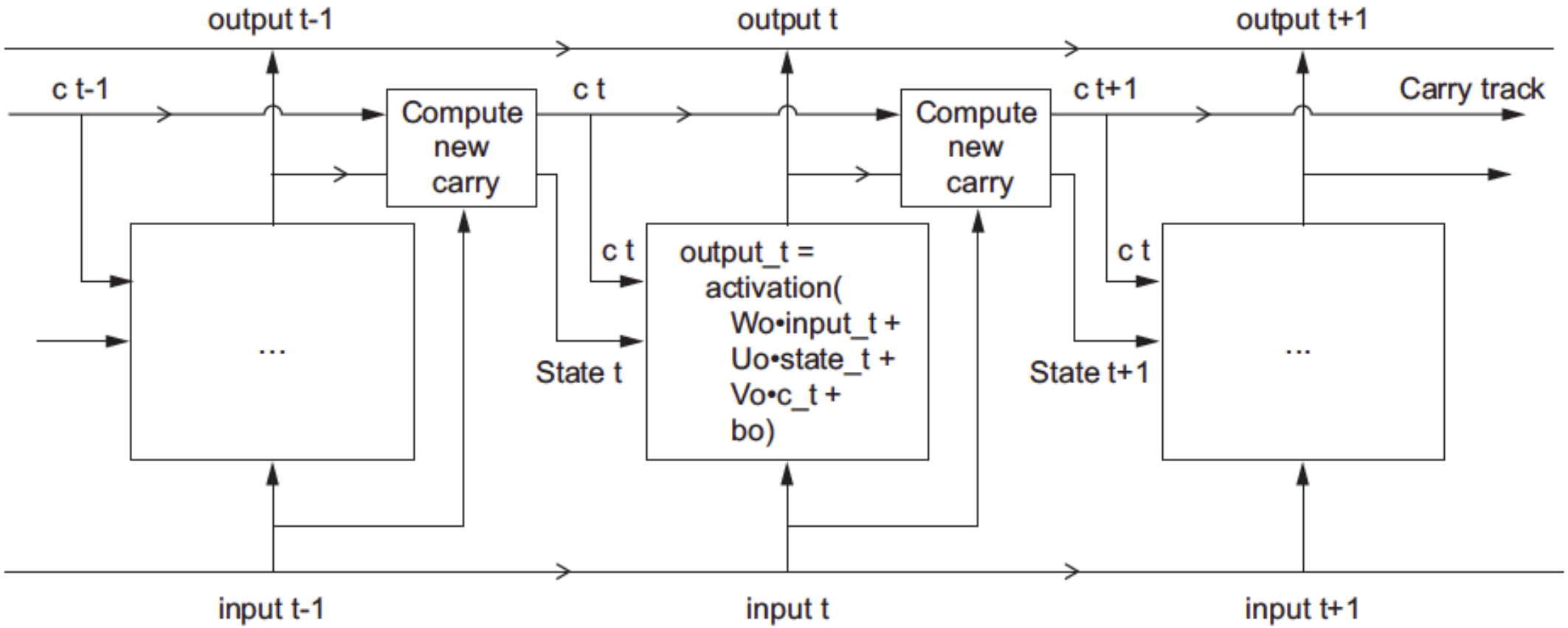


Figure 6.15 Anatomy of an LSTM

From SimpleRNN to LSTM

Listing 6.25 Pseudocode details of the LSTM architecture (1/2)

```
output_t = activation(dot(state_t, Uo) + dot(input_t, Wo) + dot(C_t, Vo) + bo)

i_t = activation(dot(state_t, Ui) + dot(input_t, Wi) + bi)
f_t = activation(dot(state_t, Uf) + dot(input_t, Wf) + bf)
k_t = activation(dot(state_t, Uk) + dot(input_t, Wk) + bk)
```

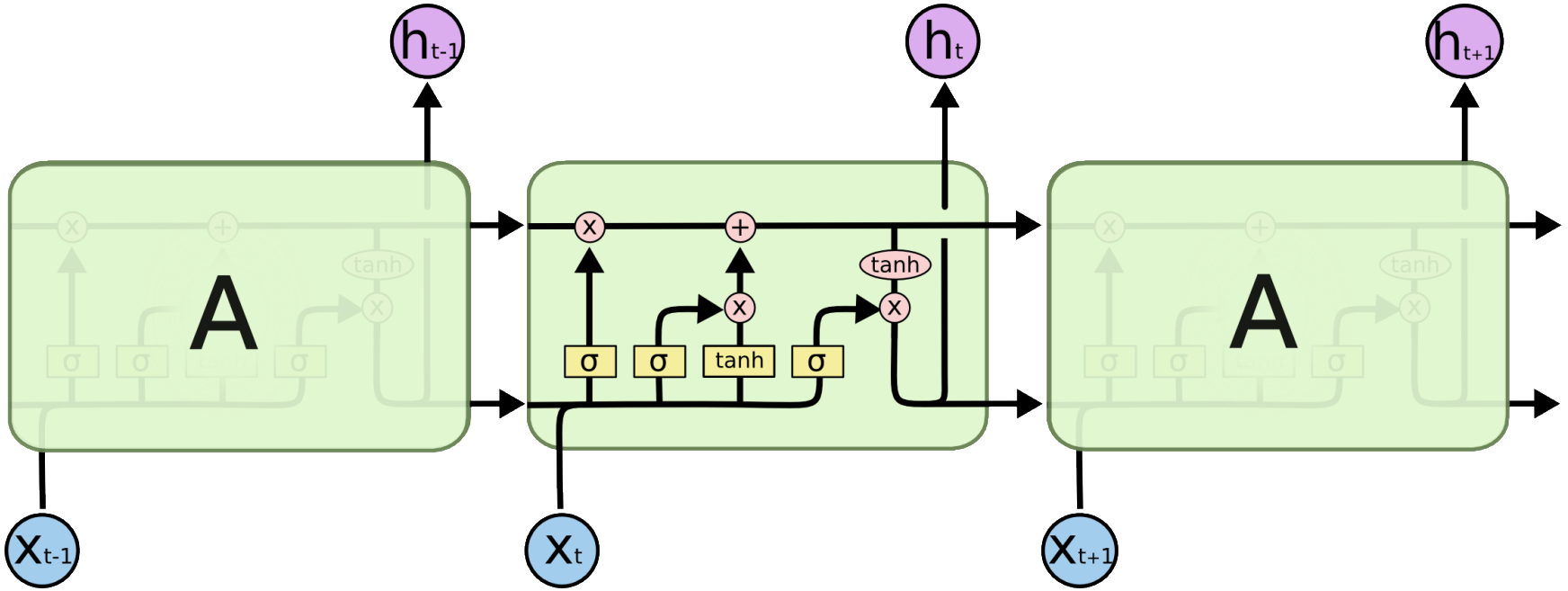
You obtain the new carry state (the next `c_t`) by combining `i_t`, `f_t`, and `k_t`.

Listing 6.26 Pseudocode details of the LSTM architecture (2/2)

```
c_t+1 = i_t * k_t + c_t * f_t
```

LSTMs: a deeper look

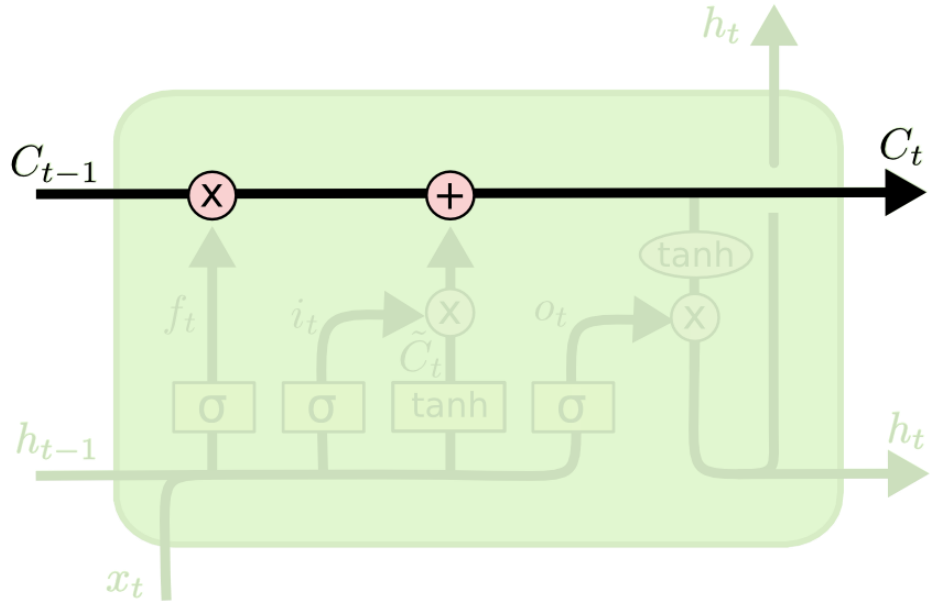
http://colah.github.io/posts/2015-08-Understanding-LSTMs/



LSTMs

Cell state

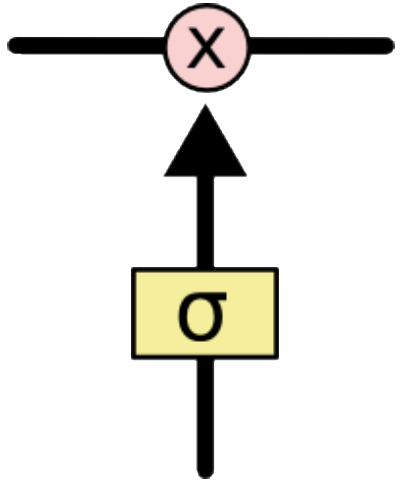
- kind of like a conveyor belt



It runs straight down the entire chain,
with only some minor linear interactions.

It's very easy for information to just flow along it unchanged.

LSTMs



Gates

A way to optionally let information through.

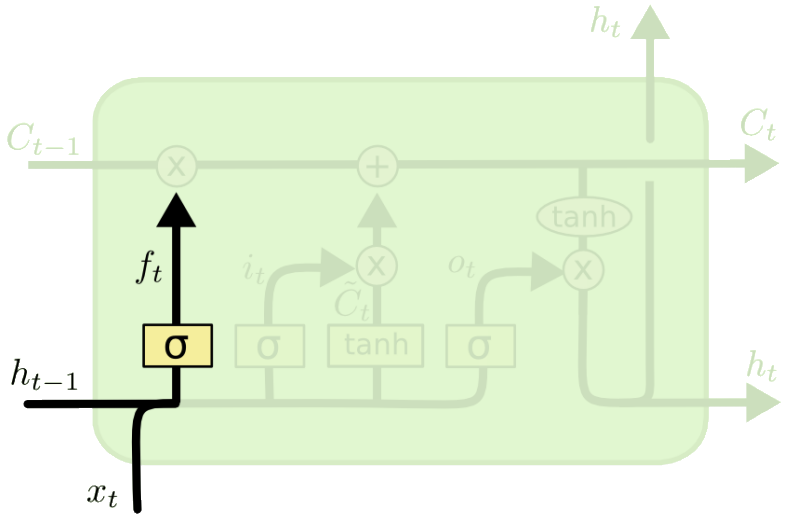
They are composed of a sigmoid neural net layer and a pointwise multiplication operation.

- A value of zero means "let nothing through," while a value of one means "let everything through!"

LSTMs

Forget gate layer

What information we're going to throw away from the cell state?



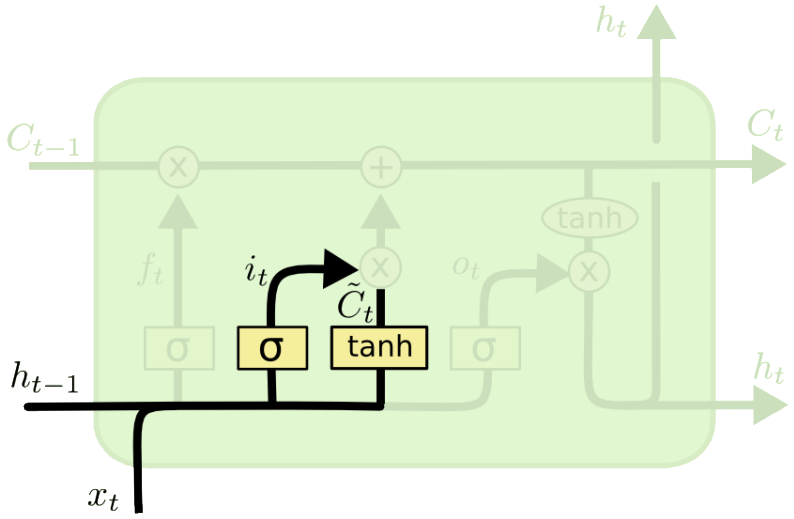
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

LSTMs

What new information we're going to store in the cell state?

Two parts:

1. A sigmoid layer called the *input gate layer* decides which values we'll update.
2. A tanh layer creates a vector of new candidate values, that could be added to the state.

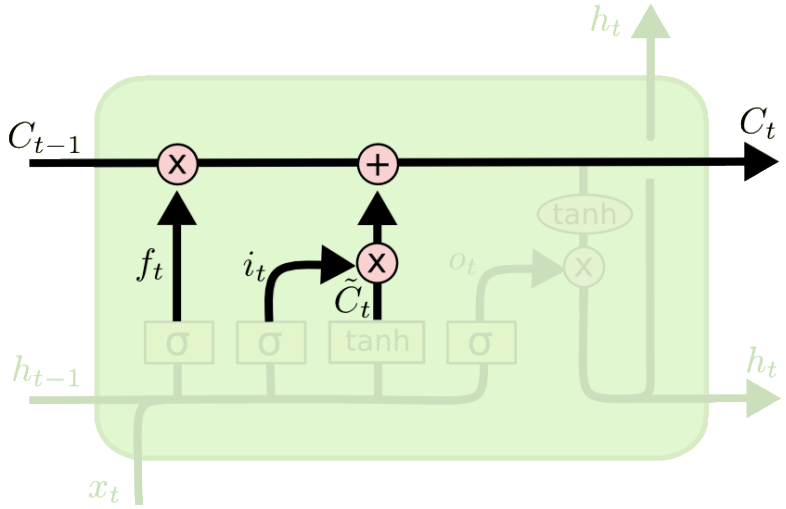


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

LSTMs

It's now time to update the old cell state C_{t-1} to C_t

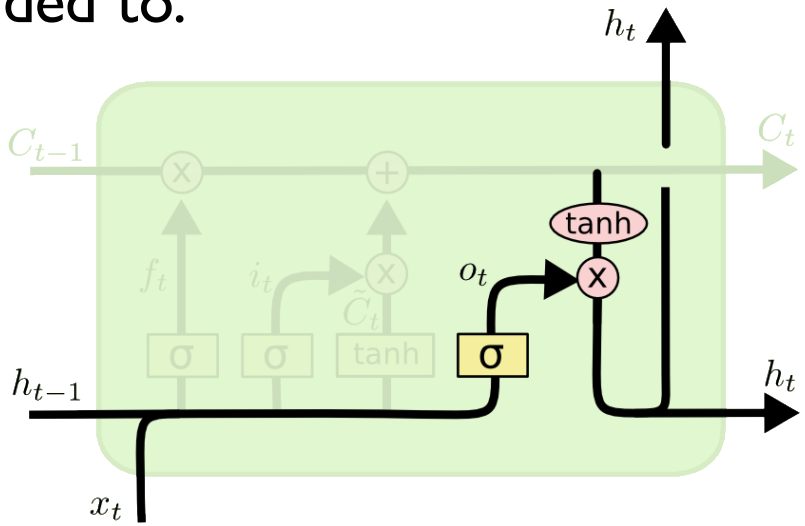


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTMs

Finally, we need to decide what we're going to output. This output will be based on our cell state but will be a filtered version.

1. A sigmoid layer to decide what parts of the cell state to output.
2. Put the cell state through tanh (to push the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

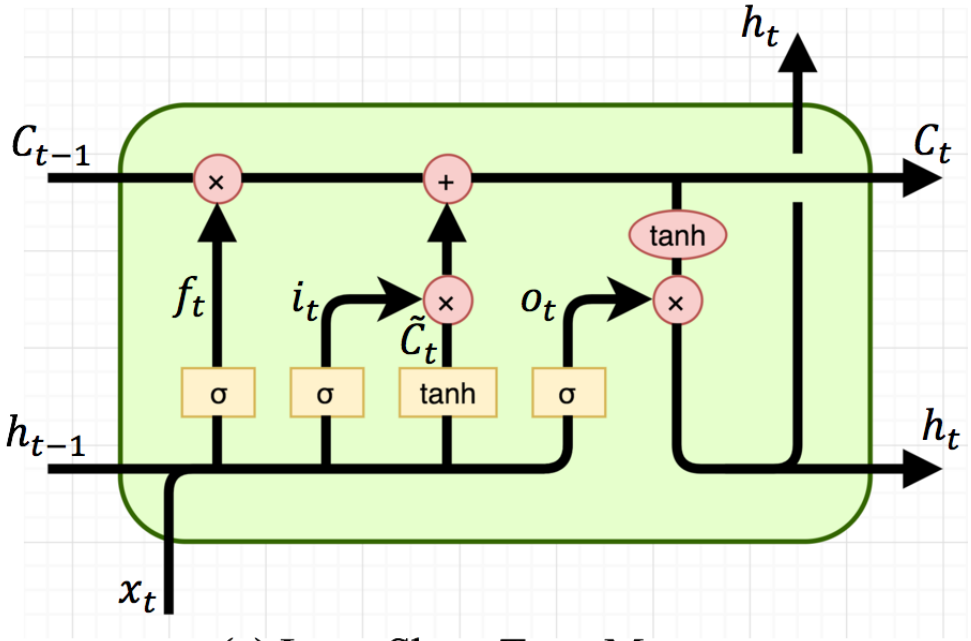


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

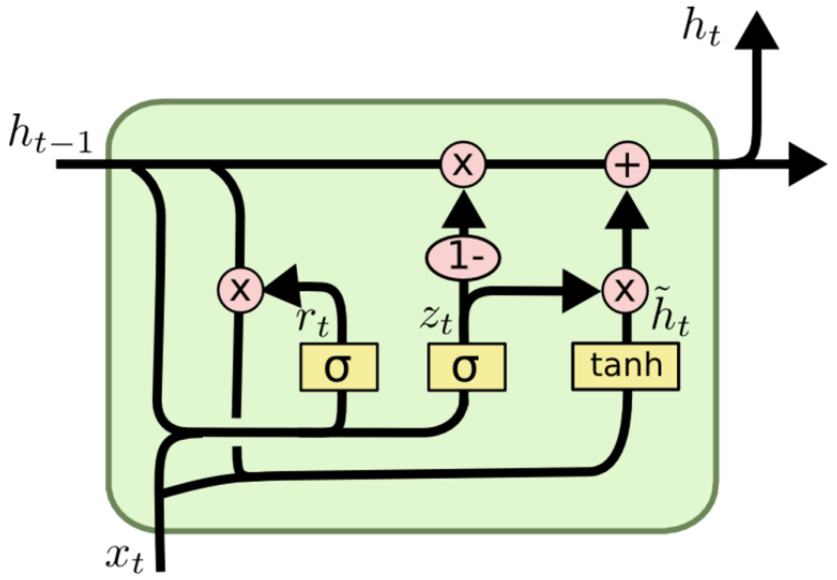
$$h_t = o_t * \tanh\left(C_t\right)$$

GRU: Gated Recurrent Units

Kyunghyun Cho et al (2014)



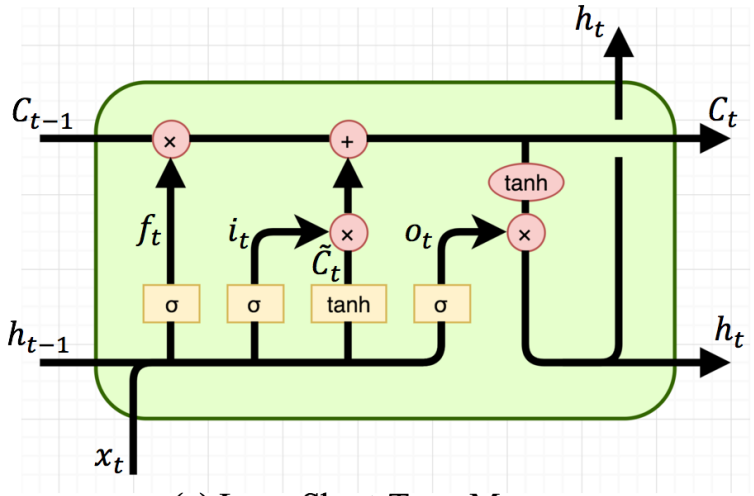
(a) Long Short-Term Memory



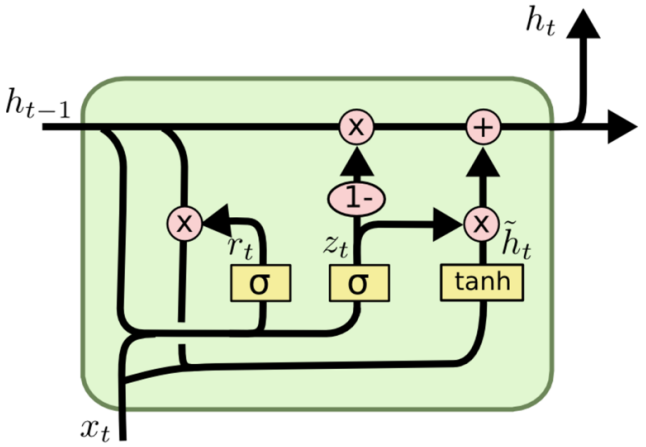
(b) Gated Recurrent Unit

https://isaacchanghau.github.io/post/lstm-gru-formula/

GRU: Gated Recurrent Units



(a) Long Short-Term Memory



(b) Gated Recurrent Unit

$$
\begin{aligned}
i_t &= \sigma\left(x_t U^i + h_{t-1} W^i\right) \\
f_t &= \sigma\left(x_t U^f + h_{t-1} W^f\right) \\
o_t &= \sigma\left(x_t U^o + h_{t-1} W^o\right) \\
\tilde{C}_t &= \tanh\left(x_t U^g + h_{t-1} W^g\right) \\
C_t &= \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right) \\
h_t &= \tanh(C_t) * o_t
\end{aligned}
$$

$$
\begin{aligned}
z_t &= \sigma\left(x_t U^z + h_{t-1} W^z\right) \\
r_t &= \sigma\left(x_t U^r + h_{t-1} W^r\right) \\
\tilde{h}_t &= \tanh\left(x_t U^h + (r_t * h_{t-1}) W^h\right) \\
h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t
\end{aligned}
$$

Transformers

"Attention is all you need"

Google, 2017

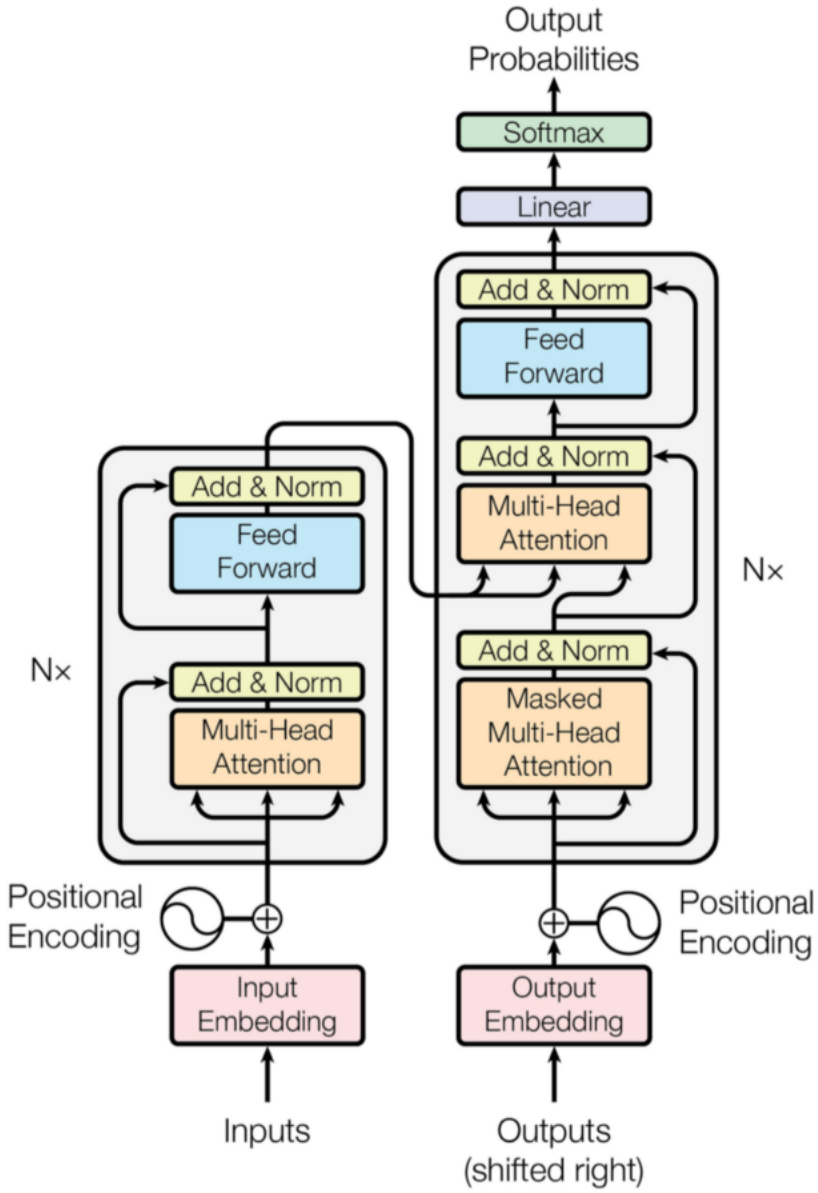


Figure 1: The Transformer - model architecture.

Progress report

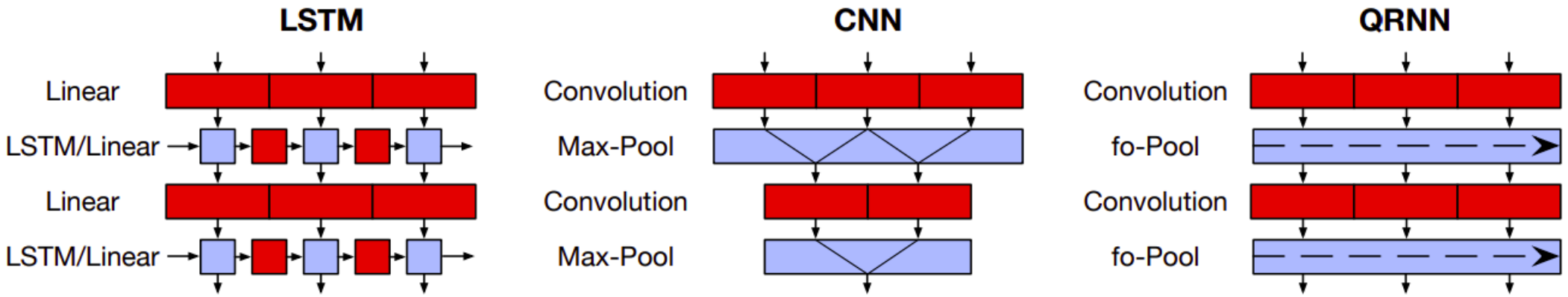
- Tuesday, 30th of Dey

- Of your project score:
  - Report 20%
  - Presentation 10%

- 10-Minute presentation

Quasi-Recurrent Neural Networks (QRNN)

Increased parallelism (16x faster than LSTMs)



Red signifies convolutions or matrix multiplications; a continuous block means that those computations can proceed in parallel. Blue signifies parameterless functions that operate in parallel along the channel/feature dimension

RNNs for sequence labeling

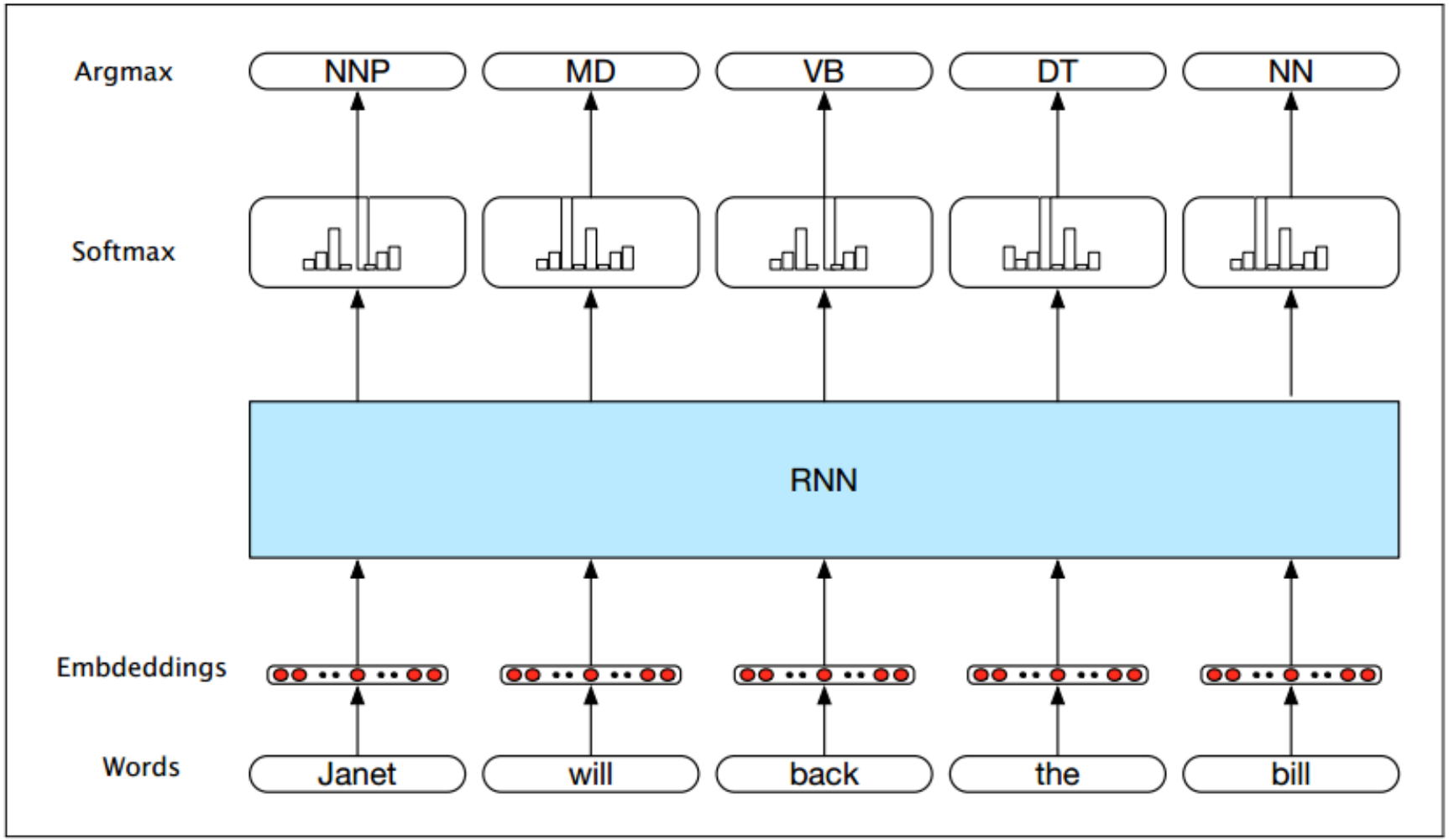


Image source

RNNs for language modeling -- training

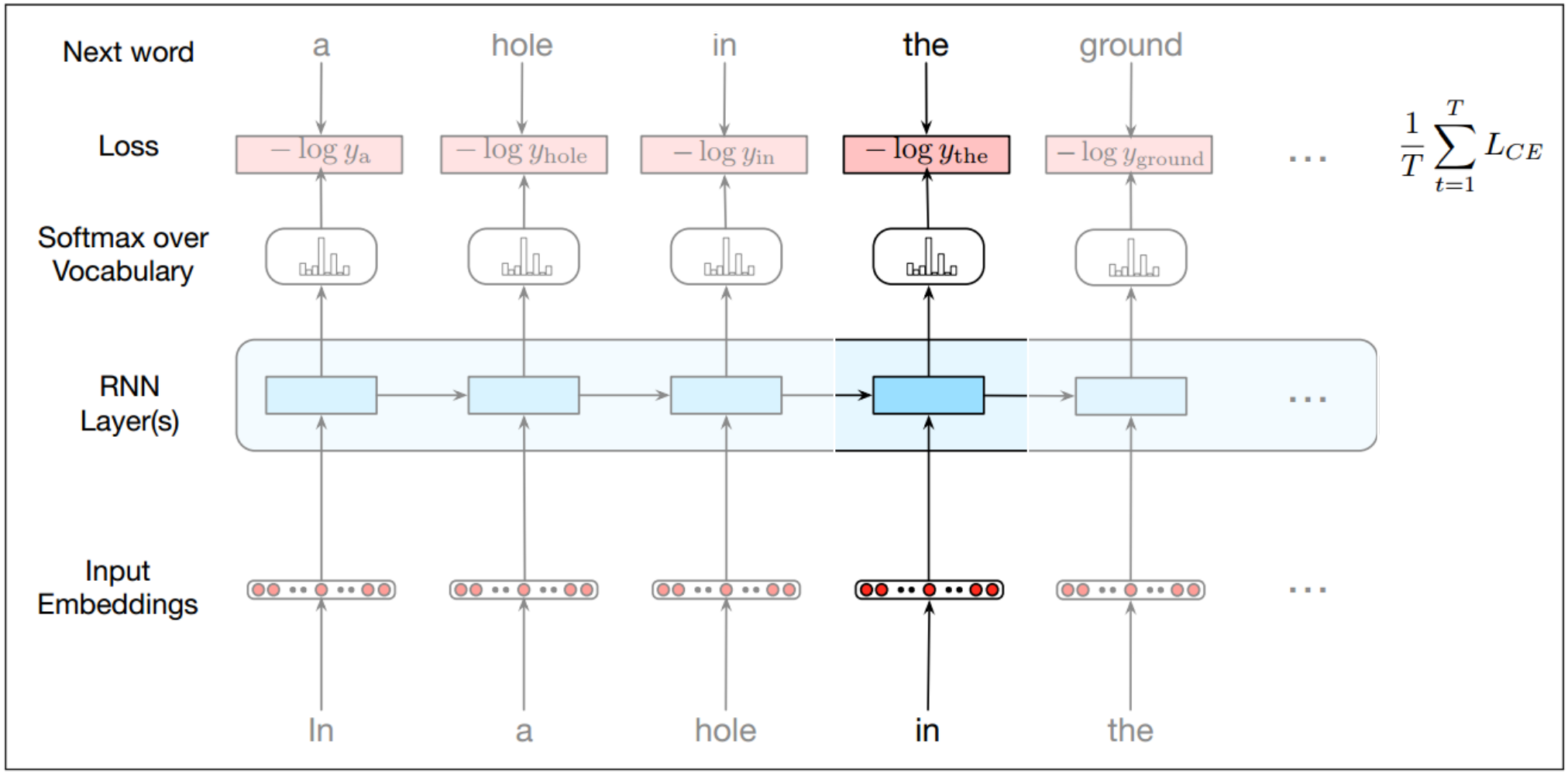


Image source

RNNs for language modeling -- autoregressive generation

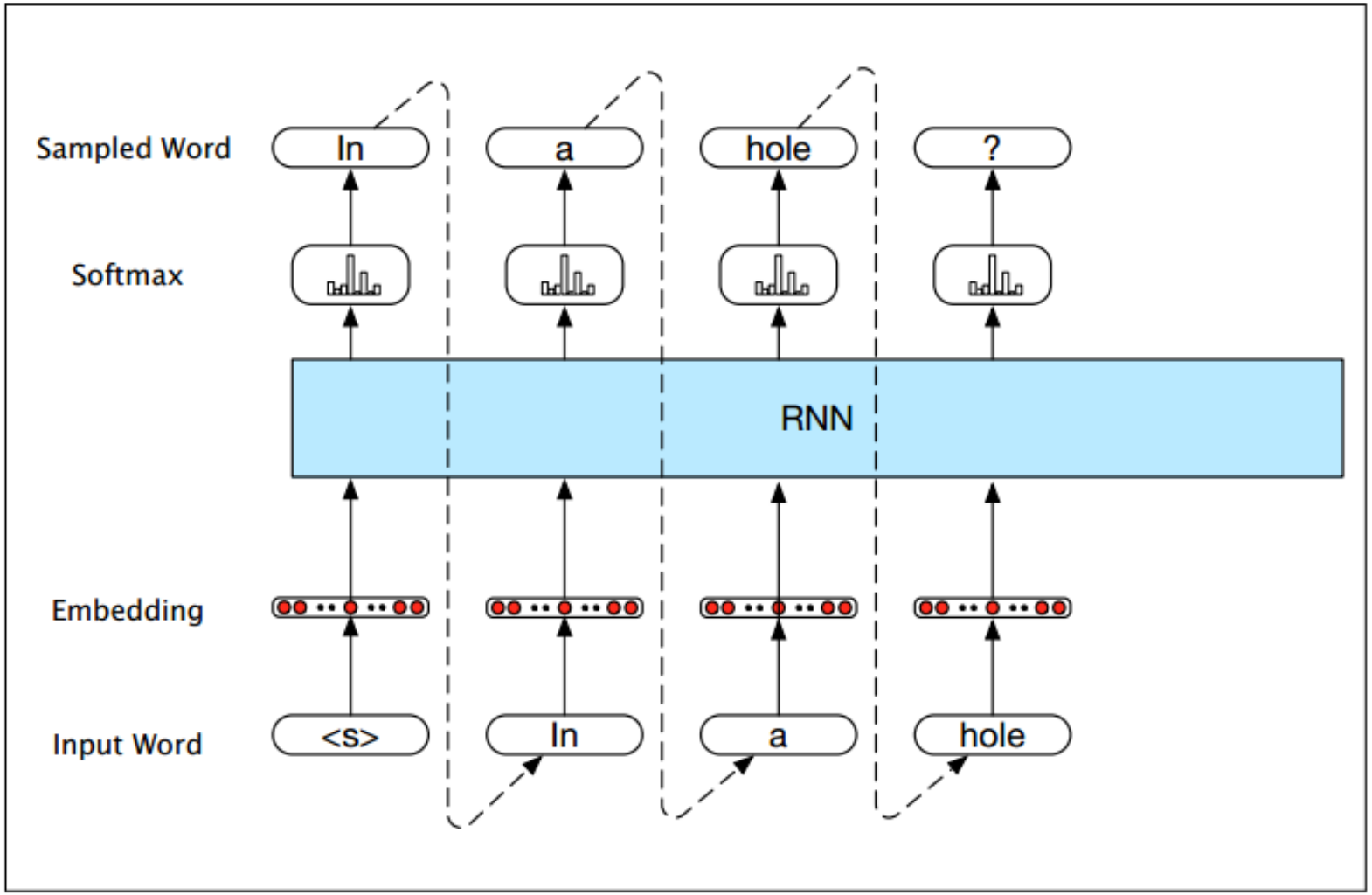
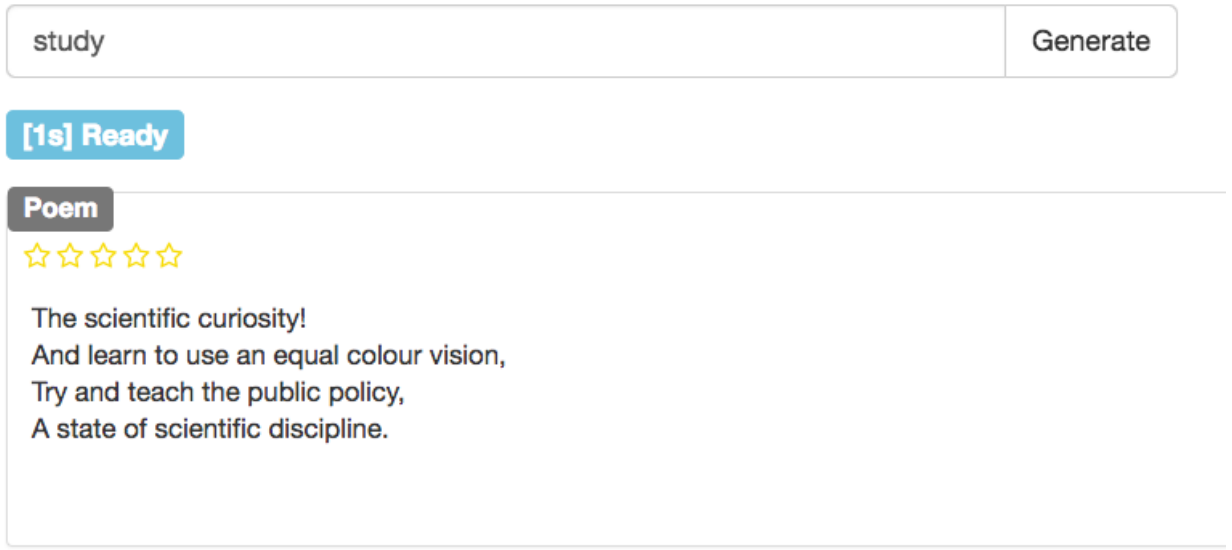


Image source

RNNs for language modeling

- Persian poetry!



روان را به دانش به کار آوری

به توران به خواب اندر آرد سری

http://52.24.230.241/poem/index.html

Afshin Khashei

RNNs for language modeling

- Persian poetry!

نیست کو معلین
دامشت خانقاه به من من از در میخانه بست
در کار ما به در این خیال ما را ببرد
ز شاه ساهد می و زلف تو را تا برسد
تا به درد تو به میخانه در آب است
که با ز من آباد به در خسابات
به خاک دل بر ت

دید او را اندر آمد ای جوان
گفت می‌دانم امیران را به جان
در دم و زشتی که افزون‌تر بود
چون تو این گفت این جهان خالی بود
گر تو ما آن را کجا زید خداست
گفت حقست آن دم آن هدیه‌هاست
کیمیا بهتر از آن زاری شوی
قبله و قدرت از آن علت شوی
بازیی زین بندها قربان شود
گر نشاید خانه را پستی شود
تا بود کار دگر خندان کند
که پذیرای خدا پیدا کند
یا چو او گردان همی‌گویید هین
در پشیمانی در آید بر زمین

سواران گردنکشان دسته دید
خردمند را او بدان خسته دید
سکندر نگه کرد پس پهلوان
به بدخواه شد شاد و روشن روان
سپاه اندر آمد به پیش سوار
خردمند و شایسته‌ی کارزار
بفرمود تا بنده آگاه دید
چنین تا بر شاه ایران کشید
نهادند چیزی که پوشیده بود
جهان را درم داد و دینار بود
سران افسر از گوهر شاهوار
نخست آفرین کرد بر کردگار
چو بهرام بشنید گریان شدند

Moghaddam and Panahbehagh, 2018
https://www.darbare.com/Post/30084

RNNs for language modeling

- Wikipedia articles

```
Naturalism and decision for the majority of Arab countries' capitalide was grounded
by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated
with Guangzham's sovereignty. His generals were the powerful ruler of the Portugal
in the [[Protestant Immineners]], which could be said to be directly in Cantonese
Communication, which followed a ceremony and set inspired prison, training. The
emperor travelled back to [[Antioch, Perth, October 25|21]] to note, the Kingdom
of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known
in western [[Scotland]], near Italy to the conquest of India with the conflict.
Copyright was the succession of independence in the slop of Syrian influence that
was a famous German movement based on a more popular servicious, non-doctrinal
and sexual power post. Many governments recognize the military housing of the
[[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]],
that is sympathetic to be to the [[Punjab Resolution]]
(PJS)[http://www.humah.yahoo.com/guardian.
cfm/7754800786d17551963s89.htm Official economics Adjoint for the Nazism, Montgomery
was swear to advance to the resources for those Socialism's rule,
was starting to signing a major tripad of aid exile.]]
```

http://karpathy.github.io

RNNs for language modeling

- Wikipedia articles

```
{ { cite journal | id=Cerling Nonforest Department|format=Newlymeslated|none } }
''www.e-complete''.

'''See also''': [[List of ethical consent processing]]

== See also ==
*[[Iender dome of the ED]]
*[[Anti-autism]]

===[[Religion|Religion]]===
*[[French Writings]]
*[[Maria]]
*[[Revelation]]
*[[Mount Agamul]]
```

http://karpathy.github.io

RNNs for language modeling

- XML

```
<page>
  <title>Antichrist</title>
  <id>865</id>
  <revision>
    <id>15900676</id>
    <timestamp>2002-08-03T18:14:12Z</timestamp>
    <contributor>
      <username>Paris</username>
      <id>23</id>
    </contributor>
    <minor />
    <comment>Automated conversion</comment>
    <text xml:space="preserve">#REDIRECT [[Christianity]]</text>
  </revision>
</page>
```

http://karpathy.github.io

RNNs for language modeling

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the compariciely in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ??. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M^\bullet} = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example ??. It may replace S by $X_{spaces,étale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma ??. Namely, by Lemma ?? we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ *(by the formal open covering* X *and a single map* $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ *over* U *compatible with the complex*

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ *is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If* T *is surjective we may assume that* T *is connected with residue fields of* S*. Moreover there exists a closed subspace* $Z \subset X$ *of* X *where* U *in* X' *is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem*

(1) f *is locally of finite type. Since* $S = \mathrm{Spec}(R)$ *and* $Y = \mathrm{Spec}(R)$.

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{\mathcal{X},\ldots,0}$.

Lemma 0.2. *Let* X *be a locally Noetherian scheme over* S, $E = \mathcal{F}_{X/S}$. *Set* $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. *Since* $\mathcal{I}^n \subset \mathcal{I}^n$ *are nonzero over* $i_0 \leq \mathfrak{p}$ *is a subset of* $\mathcal{J}_{n,0} \circ \overline{A}_2$ *works.*

Lemma 0.3. *In Situation ??. Hence we may assume* $\mathfrak{q}' = 0$.

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

Dropout for RNNs

- How to correctly apply dropout in recurrent networks isn't a trivial question.

- It has long been known that applying dropout before a recurrent layer hinders learning rather than helping with regularization.

- Yarin Gal (2015): The same dropout mask (the same pattern of dropped units) should be applied at every timestep, instead of a dropout mask that would vary randomly from timestep to timestep.

Dropout vs. recurrent dropout

```
keras.layers.LSTM(units,
                  activation='tanh',
                  dropout=0.0,
                  recurrent_dropout=0.0, …)
```

- Regular dropout masks the inputs
  - Add a `Dropout` layer after the recurrent layer if you want to mask the outputs.

- Recurrent dropout masks the connections between the recurrent units (the cell state)

Dropout for RNNs: Keras

`dropout`

- a float specifying the dropout rate for input units of the layer

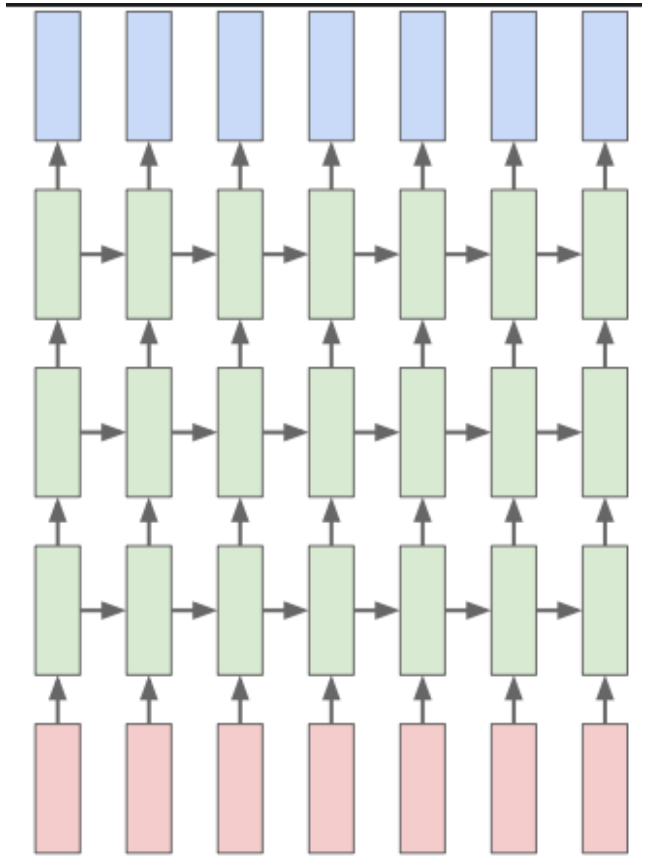
`recurrent_dropout`

- specifying the dropout rate of the recurrent state

Stacking recurrent layers

- It is a generally a good idea to increase the capacity of your network until overfitting becomes your primary obstacle (assuming that you are already taking basic steps to mitigate overfitting, such as using dropout).

- Recurrent layer stacking is a classic way to build more powerful recurrent networks:
  - For instance, what currently powers the Google translate algorithm is a stack of seven large LSTM layers -- that's huge.



Stacking recurrent layers

- To stack recurrent layers on top of each other in Keras, all intermediate layers should return their full sequence of outputs (a 3D tensor) rather than their output at the last timestep.

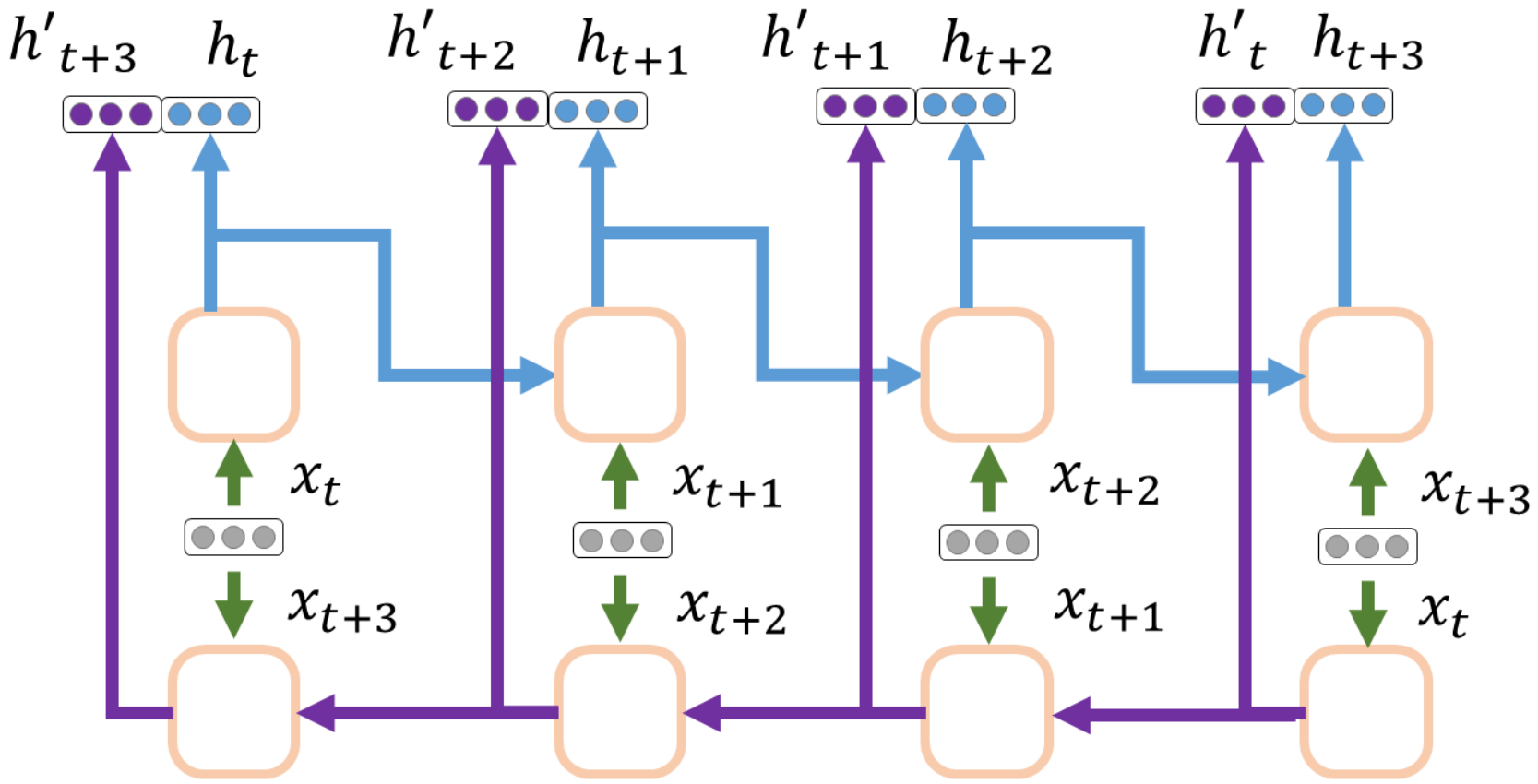
- This is done by specifying `` `return_sequences=True` ``:

Bidirectional RNNs

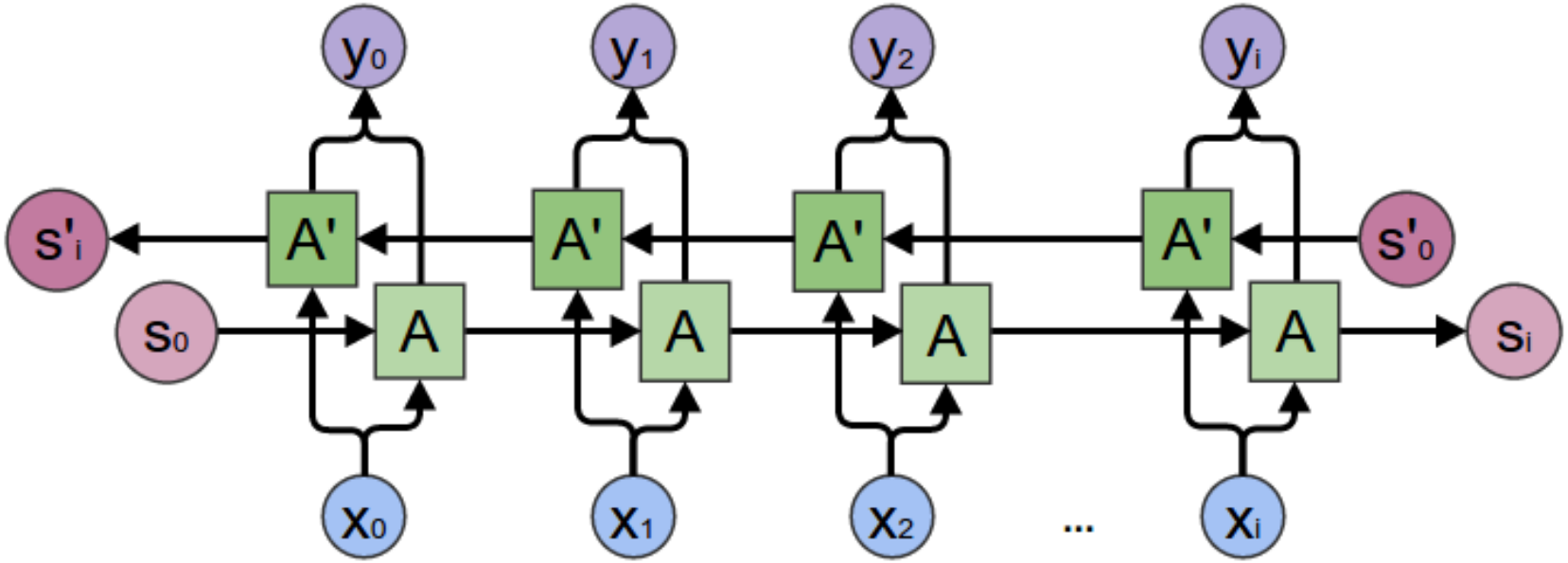
- A bidirectional RNN is common RNN variant which can offer higher performance than a regular RNN on certain tasks.

- It is frequently used in natural language processing -- you could call it the Swiss army knife of deep learning for NLP (update: not anymore).

Bidirectional RNNs



Bidirectional RNNs



http://colah.github.io

Bidirectional RNNs

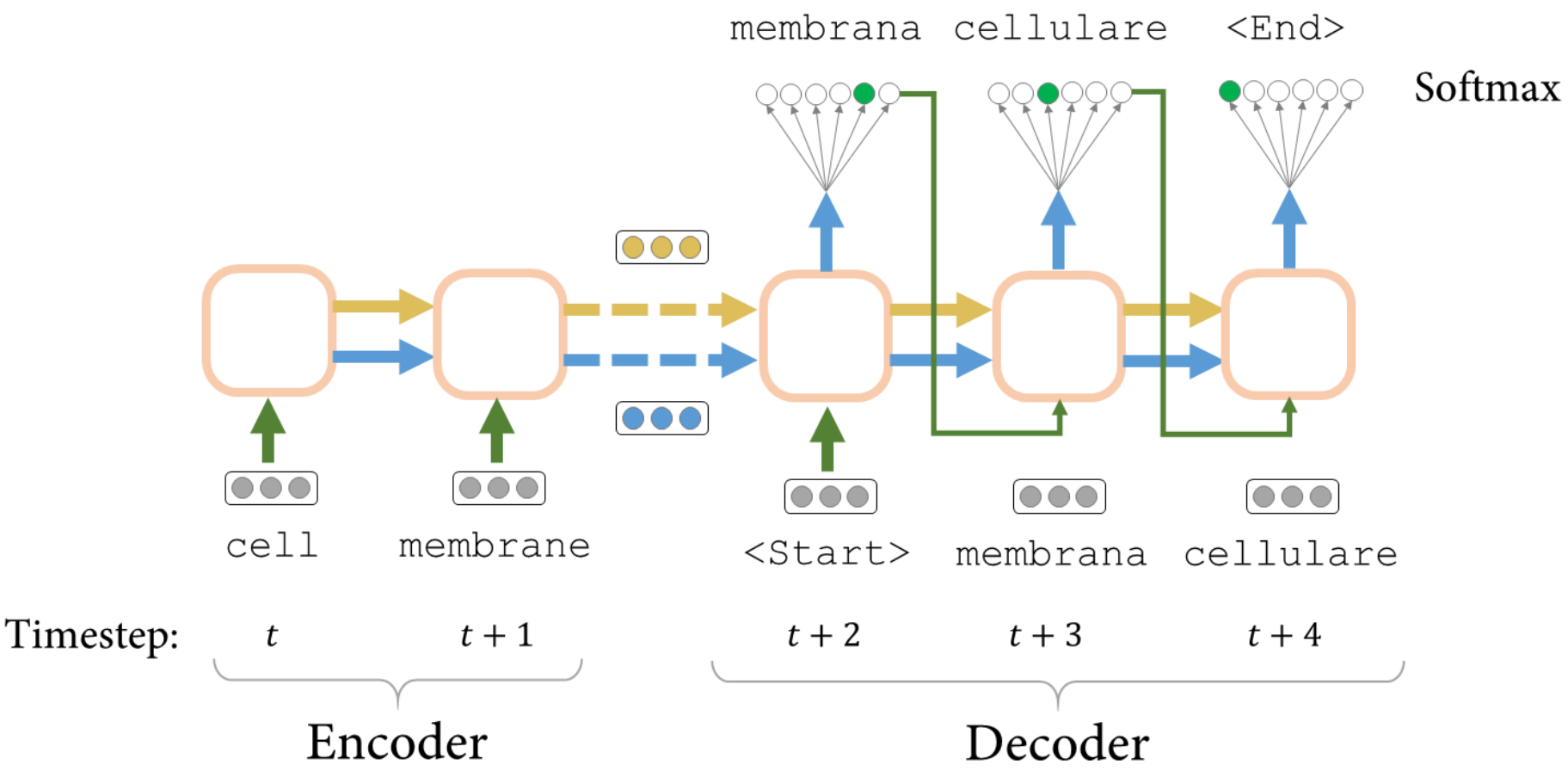
- A bidirectional RNN exploits the order-sensitivity of RNNs.
- It simply consists of two regular RNNs, such as the GRU or LSTM layers that you are already familiar with, each processing input sequence in one direction (chronologically and anti-chronologically), then merging their representations.

- By processing a sequence both way, a bidirectional RNN is able to catch patterns that may have been overlooked by a one-direction RNN.

Bidirectional RNNs

- In machine learning, representations that are **different** yet **useful** are always worth exploiting, and the more they differ the better:
    - They offer a new angle from which to look at your data, capturing aspects of the data that were missed by other approaches, and thus they can allow to boost performance on a task.
- This is the intuition behind "ensembling", a concept that we will introduce in the next chapter.

Sequence transduction (Seq2seq)



Attention

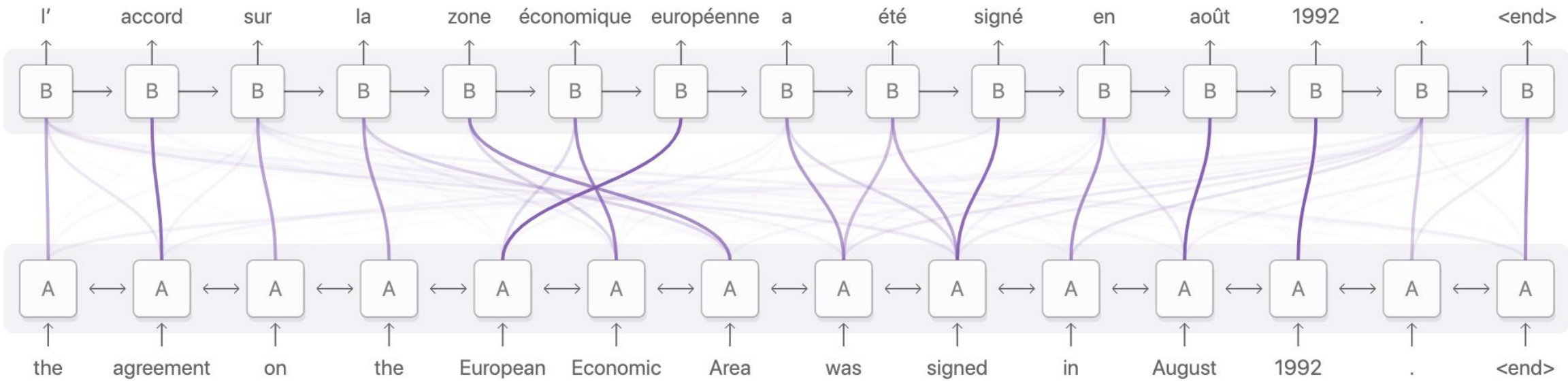
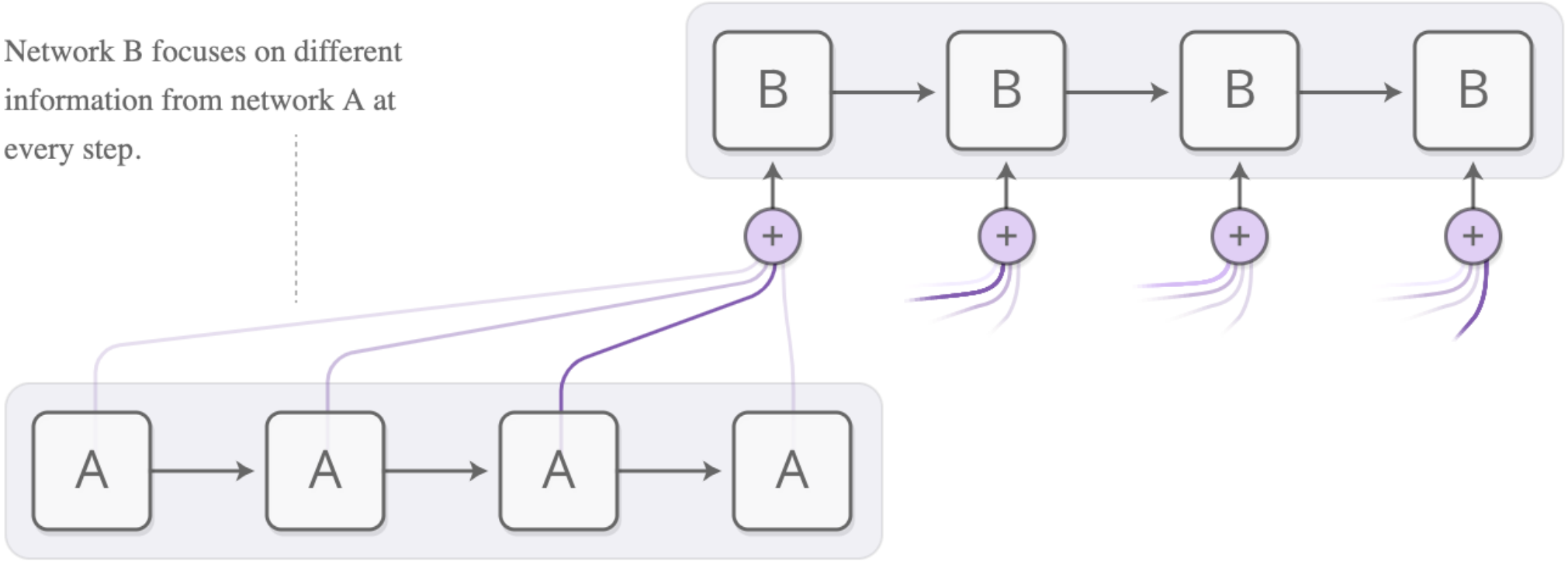


Diagram derived from Fig. 3 of Bahdanau, *et al.* 2014

https://distill.pub/2016/augmented-rnns/

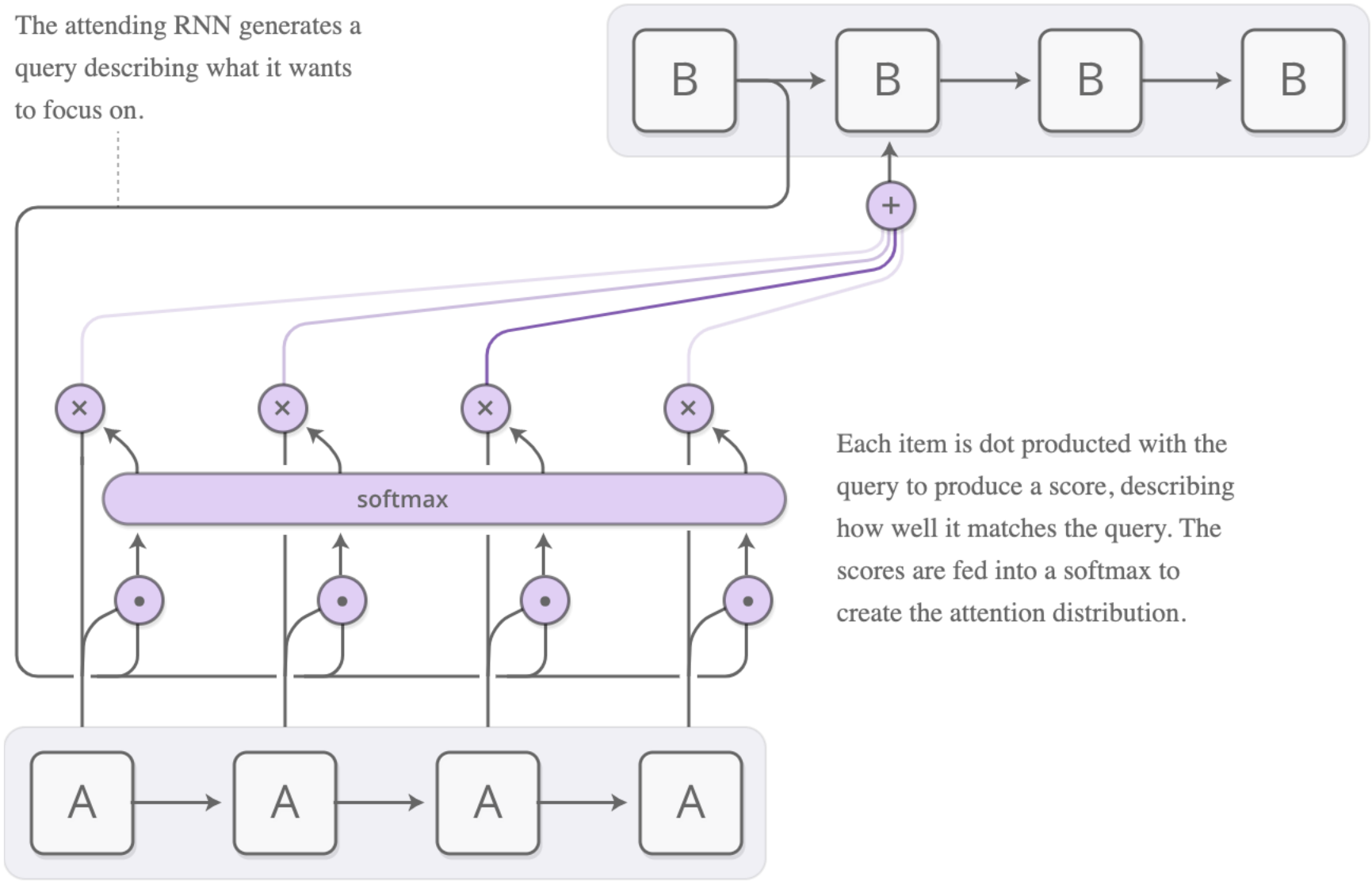
Attention

Idea: while generation focus on part of the information given

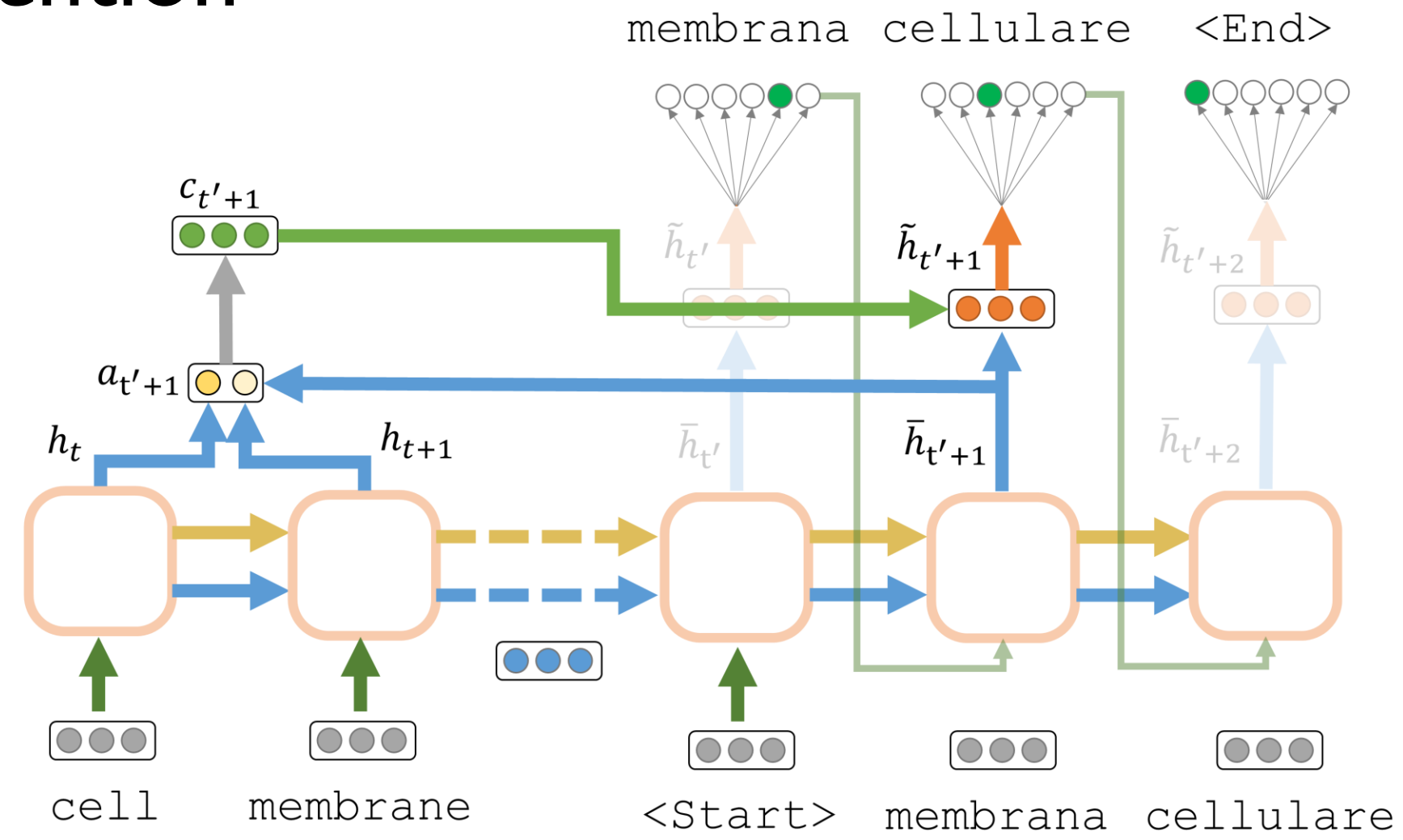


https://distill.pub/2016/augmented-rnns/

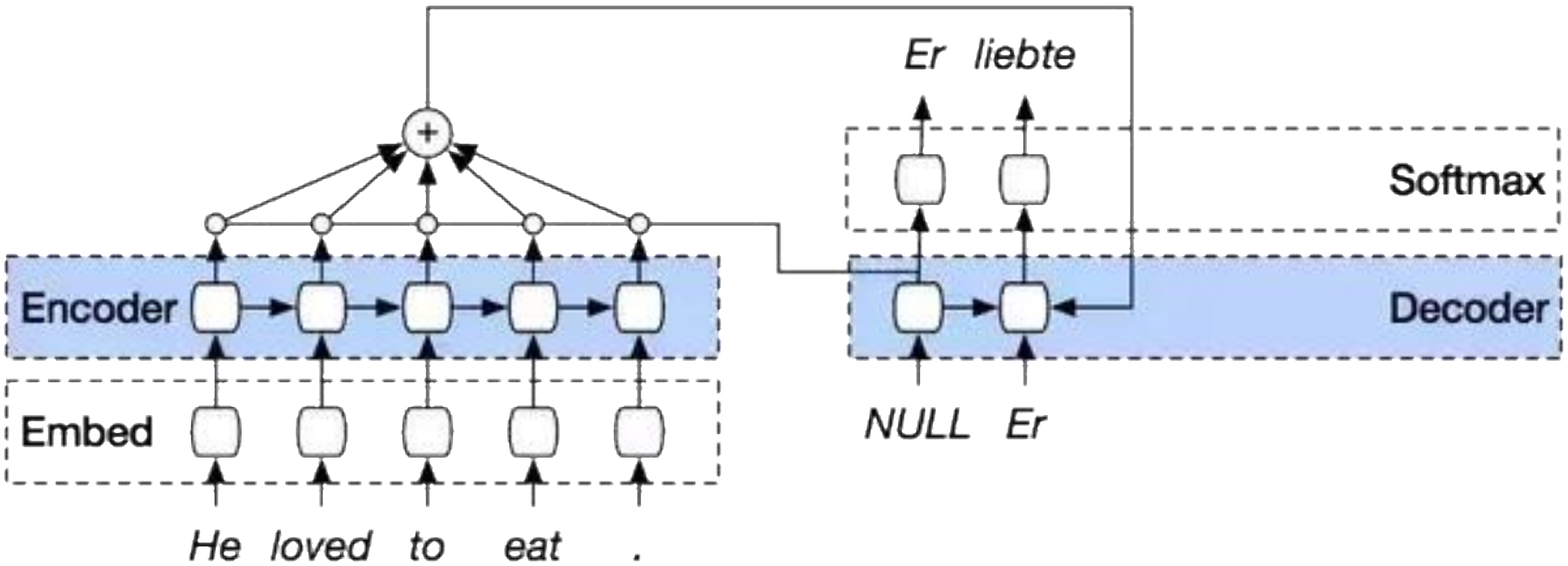
Attention


The attending RNN generates a query describing what it wants to focus on.
B
B
B
B
+
×
×
×
×
softmax
A
A
A
A
Each item is dot producted with the query to produce a score, describing how well it matches the query. The scores are fed into a softmax to create the attention distribution.

Attention

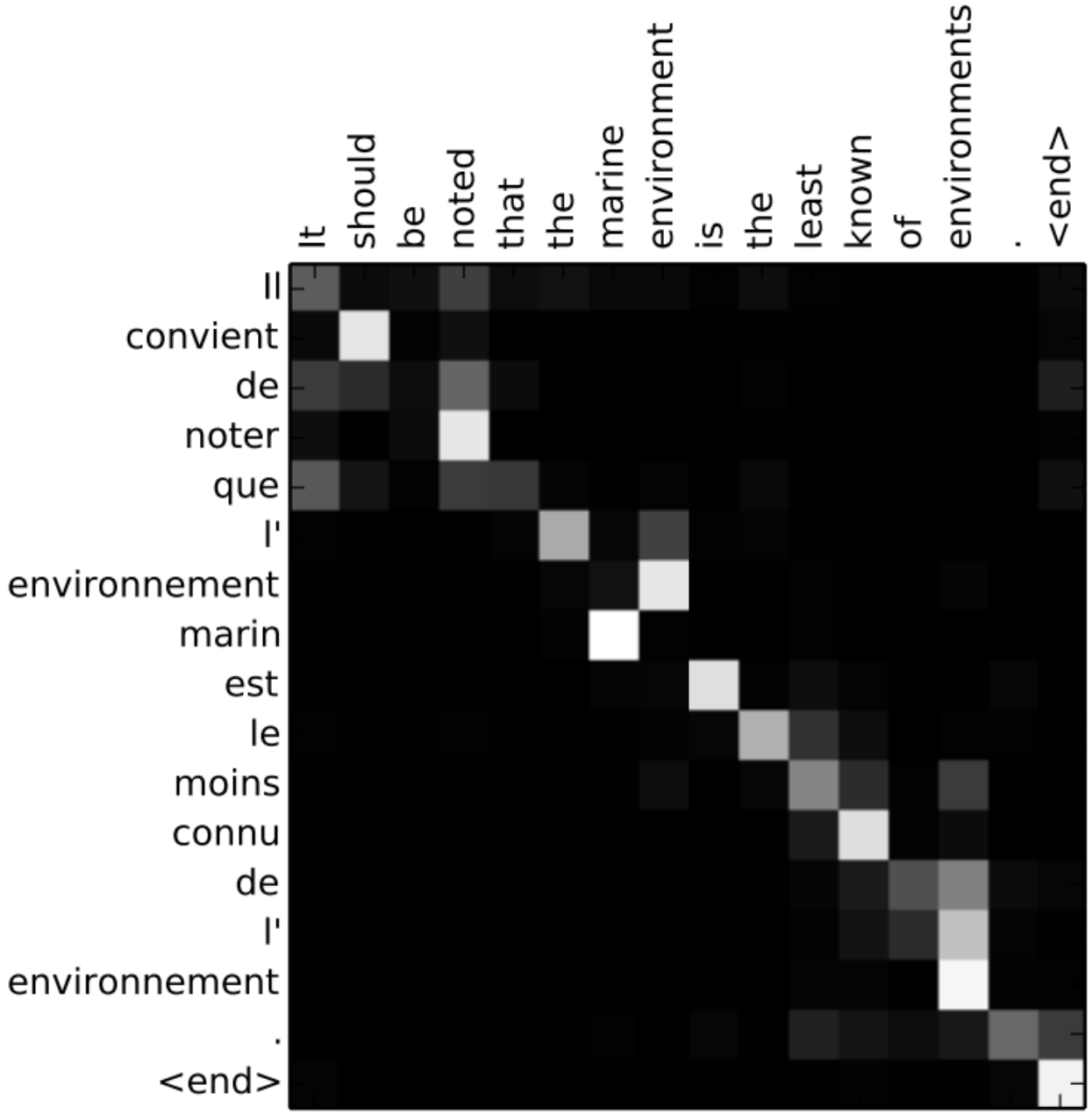

membrana
cellulare
<End>
$c_{t'+1}$
$\tilde{h}_{t'}$
$\tilde{h}_{t'+1}$
$\tilde{h}_{t'+2}$
$a_{t'+1}$
h_t
h_{t+1}
$\bar{h}_{t'}$
$\bar{h}_{t'+1}$
$\bar{h}_{t'+2}$
cell
membrane
<Start>
membrana
cellulare

Attention


Er liebte
Softmax
Encoder
Decoder
Embed
NULL Er
He loved to eat .

Attention

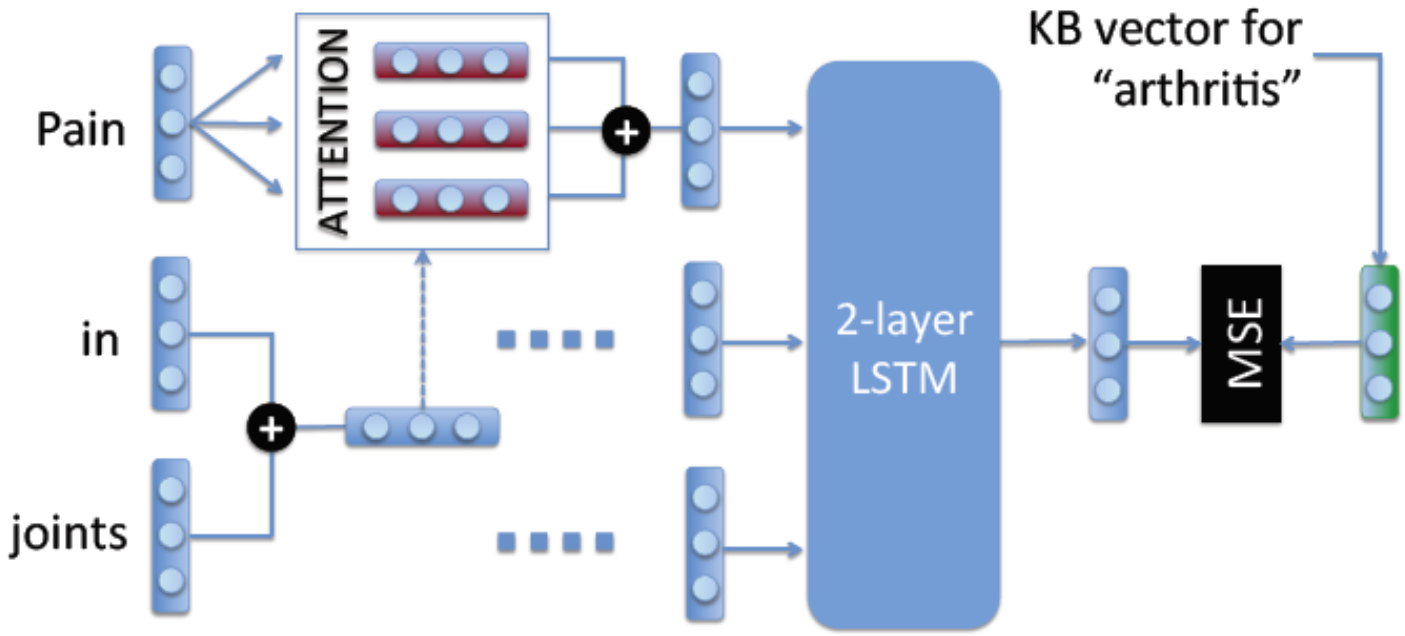
Inherent alignment



Dzmitry Bahdanau et al, 2014

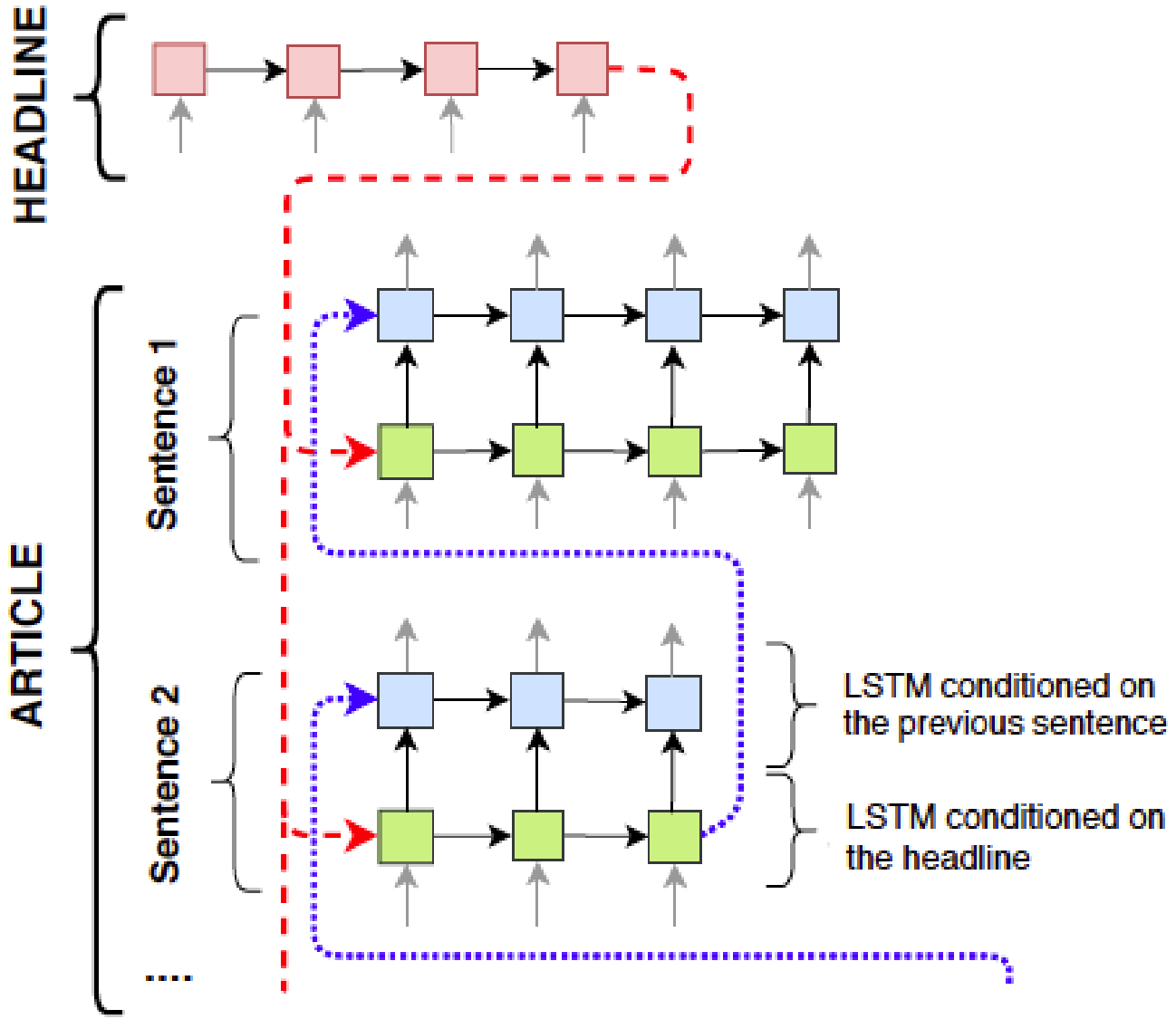
Attention

Multi-sense LSTM



Attention

Veracity detection



Attention

A stop sign is on a road with a mountain in the background.

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

Show, attend, and tell, 2015.

RNNs: sequence bucketing

Too much padding can result in reduced performance

Place sequences with similar lengths in the same batch

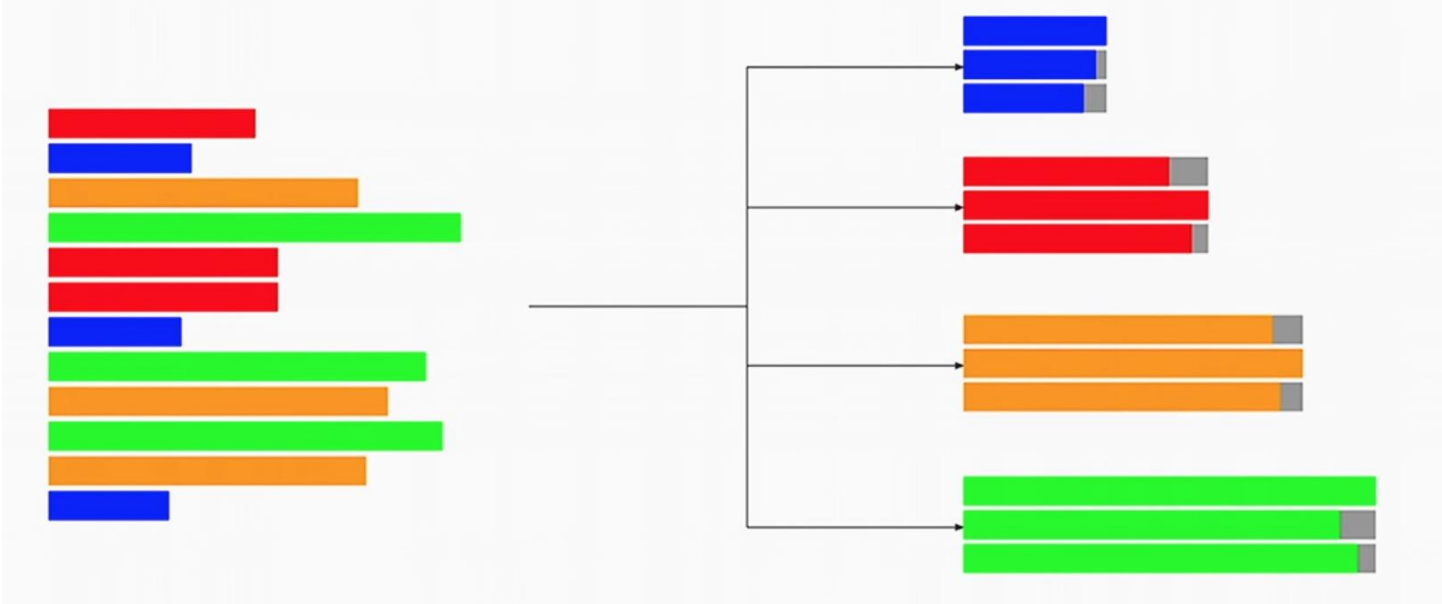


Image source

Sequence processing using ConvNets

- The same properties that make convnets excel at computer vision also make them highly relevant to sequence processing.

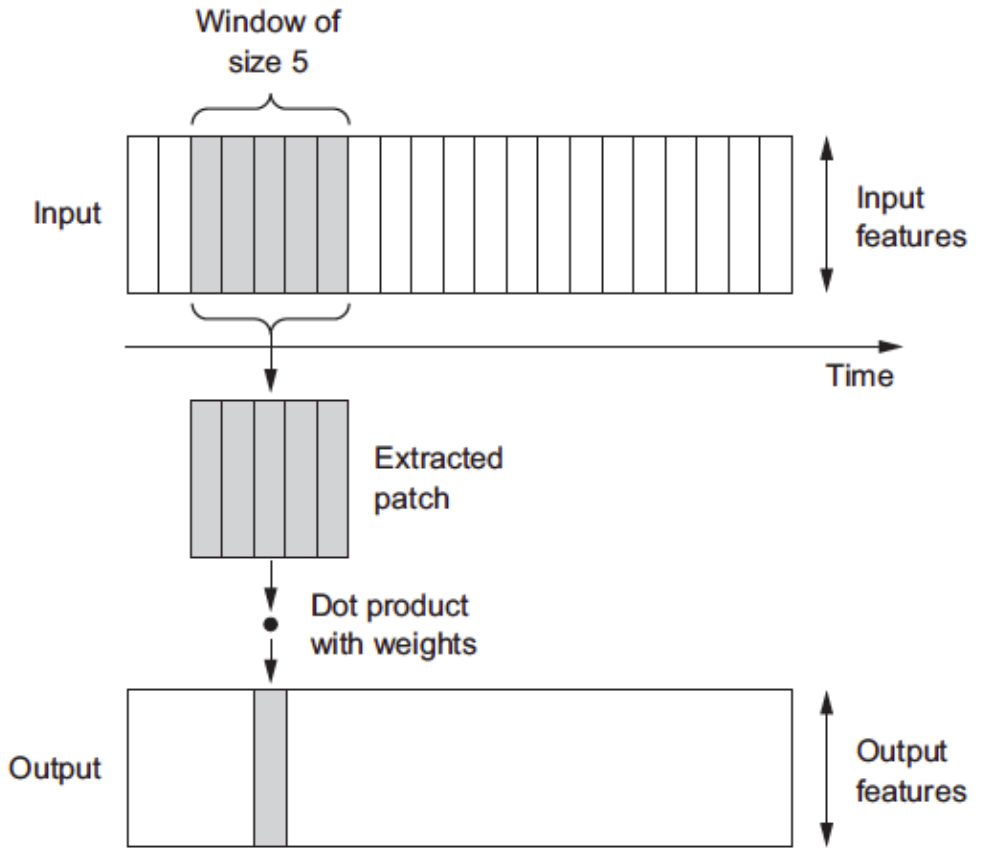
- Time can be treated as a spatial dimension, like the height or width of a 2D image.

- Such 1D convnets can be competitive with RNNs on certain sequence-processing problems, usually at a considerably cheaper computational cost.

Understanding 1D ConvNets

- 1D convolution layers can recognize local patterns in a sequence.

- A pattern learned at a certain position in a sentence can later be recognized at a different position, making 1D convnets translation invariant.



1D pooling

- Extracts 1D patches (subsequences) from an input and outputs the maximum value (max pooling) or average value (average pooling).

- Just as with 2D convnets, this is used for reducing the length of 1D inputs (subsampling).

Combining RNNs with CNNs

- Because 1D convnets process input patches independently, they aren't sensitive to the order of the timesteps (beyond a local scale, the size of the convolution windows), unlike RNNs.

Combining RNNs with CNNs

- One strategy to combine the speed and lightness of convnets with the order-sensitivity of RNNs is to use a 1D convnet as a preprocessing step before an RNN.

- Especially beneficial when you're dealing with sequences that are so long they can't realistically be processed with RNNs, such as sequences with thousands of steps.

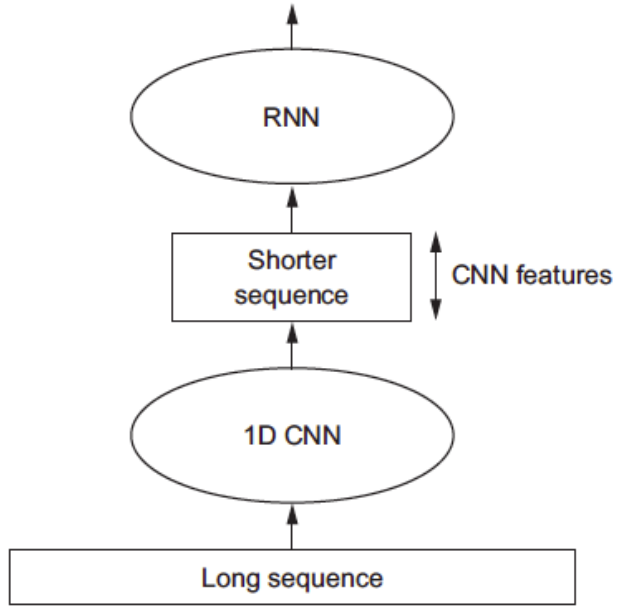


Figure 6.30 Combining a 1D convnet and an RNN for processing long sequences

Allen NLP demo - https://demo.allennlp.org/

Passage

IUST is one of Iran's major universities which is located in the north east of Tehran. Tehran University is located at the center of Tehran.

Question

where is IUST locate?

Model

- (●) ELMo-BiDAF (trained on SQuAD)
- ( ) BiDAF (trained on SQuAD)
- ( ) NAQANet (trained on DROP)

Answer

north east of Tehran

Passage Context

IUST is one of Iran's major universities which is located in the **north east of Tehran** . Tehran University is located at the center of Tehran.

Questions?