# Week 4: Embeddings and Recurrent Neural Networks

Matthew Willetts - Alexander Camuto

# Outline

Embeddings

# Outline

Embeddings

Recurrent Neural Networks

# Outline

Embeddings

Recurrent Neural Networks

Long Short-term Memory Networks

# Embeddings

# Symbolic variable

# Symbolic variable

- Text: characters, words, bigrams...

# Symbolic variable

- Text: characters, words, bigrams...
- Recommender Systems: item ids, user ids

# Symbolic variable

- Text: characters, words, bigrams...
- Recommender Systems: item ids, user ids
- Any categorical descriptor: tags, movie genres, visited URLs, skills on a resume, product categories...

# Symbolic variable

- Text: characters, words, bigrams...
- Recommender Systems: item ids, user ids
- Any categorical descriptor: tags, movie genres, visited URLs, skills on a resume, product categories...

- Notation: Symbol $s$ in vocabulary $V$

# One-hot representation

$$onehot(\text{'salad'}) = [0, 0, 1, \ldots, 0] \in \{0,1\}^{|V|}$$

# One-hot representation

$$onehot(\text{'salad'}) = [0, 0, 1, \dots, 0] \in \{0, 1\}^{|V|}$$


- Sparse, discrete, large dimension $|V|$
- Each axis has a meaning
- Symbols are equidistant from each other:

euclidean distance = $\sqrt{2}$

# Embedding

$$embedding(\text{'salad'}) = [3.28, -0.45, \ldots 7.11] \in \mathbb{R}^d$$

# Embedding

$$embedding(\text{'salad'}) = [3.28, -0.45, \ldots 7.11] \in \mathbb{R}^d$$

- Continuous and dense
- Can represent a huge vocabulary in low dimension, typically: $d \in \{16, 32, \ldots, 4096\}$
- Axis have no meaning *a priori*
- Embedding metric can capture semantic distance

# Embedding

$$embedding(\text{'salad'}) = [3.28, -0.45, \ldots 7.11] \in \mathbb{R}^d$$

- Continuous and dense
- Can represent a huge vocabulary in low dimension, typically: $d \in \{16, 32, \ldots, 4096\}$
- Axis have no meaning *a priori*
- Embedding metric can capture semantic distance

**Neural Networks compute transformations on continuous vectors**

# Implementation with Keras

Size of vocabulary $n = |V|$, size of embedding $d$

```
# input: batch of integers
Embedding(output_dim=d, input_dim=n, input_length=1)
# output: batch of float vectors
```

# Implementation with Keras

Size of vocabulary $n = |V|$, size of embedding $d$

```
# input: batch of integers
Embedding(output_dim=d, input_dim=n, input_length=1)
# output: batch of float vectors
```

- Equivalent to one-hot encoding multiplied by a weight matrix $\mathbf{W} \in \mathbb{R}^{n \times d}$:

$$embedding(x) = onehot(x) . \mathbf{W}$$

# Implementation with Keras

Size of vocabulary $n = |V|$, size of embedding $d$

```
# input: batch of integers
Embedding(output_dim=d, input_dim=n, input_length=1)
# output: batch of float vectors
```

- Equivalent to one-hot encoding multiplied by a weight matrix $\mathbf{W} \in \mathbb{R}^{n \times d}$:

$$embedding(x) = onehot(x).\mathbf{W}$$

- $\mathbf{W}$ is typically **randomly initialized**, then **tuned by backprop**

# Implementation with Keras

Size of vocabulary $n = |V|$, size of embedding $d$

```
# input: batch of integers
Embedding(output_dim=d, input_dim=n, input_length=1)
# output: batch of float vectors
```

- Equivalent to one-hot encoding multiplied by a weight matrix $\mathbf{W} \in \mathbb{R}^{n \times d}$:

$$embedding(x) = onehot(x).\,\mathbf{W}$$

- $\mathbf{W}$ is typically **randomly initialized**, then **tuned by backprop**
- $\mathbf{W}$ are trainable parameters of the model

# Distance and similarity in Embedding space

## Euclidean distance

$d(x, y) = ||x - y||_2$

- Simple with good properties
- Dependent on norm (embeddings usually unconstrained)

# Distance and similarity in Embedding space

## Euclidean distance

$d(x, y) = ||x - y||_2$

- Simple with good properties
- Dependent on norm (embeddings usually unconstrained)

## Cosine similarity

$cosine(x, y) = \frac{x \cdot y}{||x|| \cdot ||y||}$

- Angle between points, regardless of norm
- $\boldsymbol{cosine(x, y)} \in \boldsymbol{(-1, 1)}$
- Expected cosine similarity of random pairs of vectors is $\mathbf{0}$

# Distance and similarity in Embedding space

If $x$ and $y$ both have unit norms:

$$||x - y||_2^2 = 2 \cdot (1 - cosine(x, y))$$

# Distance and similarity in Embedding space

If $x$ and $y$ both have unit norms:

$$||x - y||_2^2 = 2 \cdot (1 - cosine(x, y))$$

or alternatively:

$$cosine(x, y) = 1 - \frac{||x - y||_2^2}{2}$$

# Distance and similarity in Embedding space

If $x$ and $y$ both have unit norms:

$$||x - y||_2^2 = 2 \cdot (1 - cosine(x, y))$$

or alternatively:

$$cosine(x, y) = 1 - \frac{||x - y||_2^2}{2}$$

Alternatively, dot product (unnormalized) is used in practice as a pseudo similarity

# Visualizing Embeddings

- Visualizing requires a projection in 2 or 3 dimensions
- Objective: visualize which embedded symbols are similar

# Visualizing Embeddings

- Visualizing requires a projection in 2 or 3 dimensions
- Objective: visualize which embedded symbols are similar

PCA

- Limited by linear projection, embeddings usually have complex high dimensional structure

# Visualizing Embeddings

- Visualizing requires a projection in 2 or 3 dimensions
- Objective: visualize which embedded symbols are similar

## PCA

- Limited by linear projection, embeddings usually have complex high dimensional structure

## t-SNE

Visualizing data using t-SNE, L van der Maaten, G Hinton, *The Journal of Machine Learning Research*, 2008

# t-Distributed Stochastic Neighbor Embedding

- Unsupervised, low-dimension, non-linear projection
- Optimized to preserve relative distances between nearest neighbors
- Global layout is not necessarily meaningful

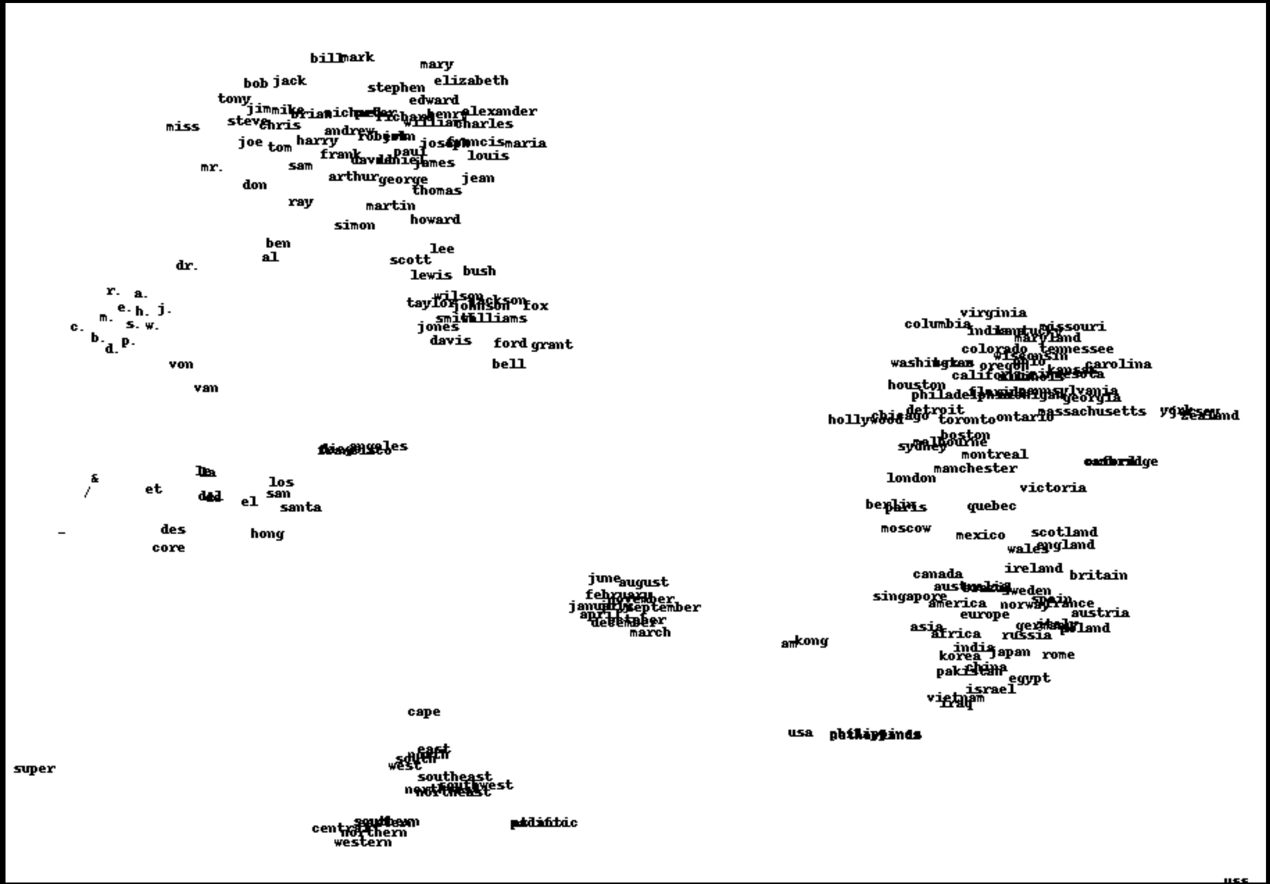
# t-Distributed Stochastic Neighbor Embedding

- Unsupervised, low-dimension, non-linear projection
- Optimized to preserve relative distances between nearest neighbors
- Global layout is not necessarily meaningful

t-SNE projection is non deterministic (depends on initialization)

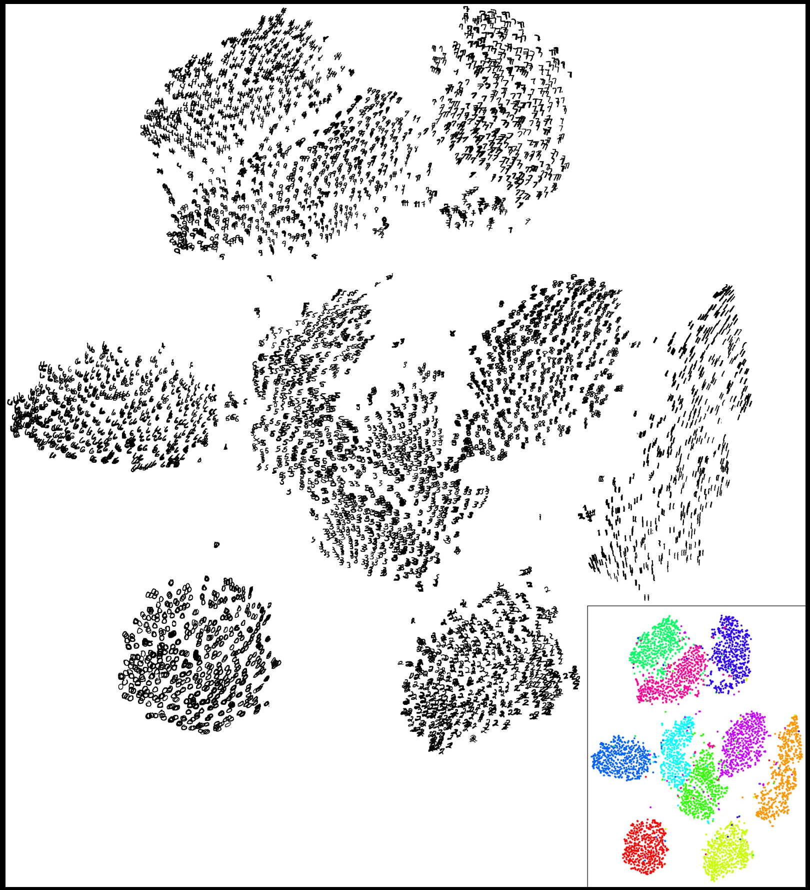
- Critical parameter: perplexity, usually set to 20, 30
- See [http://distill.pub/2016/misread-tsne/](http://distill.pub/2016/misread-tsne/)

# Example word vectors




excerpt from work by J. Turian on a model trained by R. Collobert et al. 2008

# Visualizing Mnist



# Take Away on Embeddings

**For text applications, inputs of Neural Networks are Embeddings**

# Take Away on Embeddings

**For text applications, inputs of Neural Networks are Embeddings**

- If **little training data** and a wide vocabulary not well covered by training data, use **pre-trained embeddings** (transfer learning from Glove, word2vec or fastText embeddings)

# Take Away on Embeddings

**For text applications, inputs of Neural Networks are Embeddings**

- If **little training data** and a wide vocabulary not well covered by training data, use **pre-trained embeddings** (transfer learning from Glove, word2vec or fastText embeddings)
- If **large training data** with labels, directly learn task-specific embedding with tools such as **fastText in supervised mode**.

# Take Away on Embeddings

**For text applications, inputs of Neural Networks are Embeddings**

- If **little training data** and a wide vocabulary not well covered by training data, use **pre-trained embeddings** (transfer learning from Glove, word2vec or fastText embeddings)
- If **large training data** with labels, directly learn task-specific embedding with tools such as **fastText in supervised mode**.
- These methods use **Bag-of-Words** (BoW): they **ignore the order** in word sequences

# Take Away on Embeddings

**For text applications, inputs of Neural Networks are Embeddings**

- If **little training data** and a wide vocabulary not well covered by training data, use **pre-trained embeddings** (transfer learning from Glove, word2vec or fastText embeddings)
- If **large training data** with labels, directly learn task-specific embedding with tools such as **fastText in supervised mode**.
- These methods use **Bag-of-Words** (BoW): they **ignore the order** in word sequences
- Depth & non-linear activations on hidden layers are not that useful for BoW text classification.

# Take Away on Embeddings

**For text applications, inputs of Neural Networks are Embeddings**

- If **little training data** and a wide vocabulary not well covered by training data, use **pre-trained embeddings** (transfer learning from Glove, word2vec or fastText embeddings)
- If **large training data** with labels, directly learn task-specific embedding with tools such as **fastText in supervised mode**.
- These methods use **Bag-of-Words** (BoW): they **ignore the order** in word sequences
- Depth & non-linear activations on hidden layers are not that useful for BoW text classification.

**Deep Learning in NLP** has recently caught up with other domains such as computer vision and speech recognition - see Tranformer Networks like GPT and BERT

# Language Modelling and Recurrent Neural Networks

# Language Models

Assign a probability to a sequence of words, such that plausible sequences have higher probabilities e.g:

- $p(\text{"I like cats"}) > p(\text{"I table cats"})$
- $p(\text{"I like cats"}) > p(\text{"like I cats"})$

# Language Models

Assign a probability to a sequence of words, such that plausible sequences have higher probabilities e.g:

- $p(\text{"I like cats"}) > p(\text{"I table cats"})$
- $p(\text{"I like cats"}) > p(\text{"like I cats"})$

**Auto-regressive sequence modelling**

$p_\theta(w_0)$

$p_\theta$ is parametrized by a neural network.

# Language Models

Assign a probability to a sequence of words, such that plausible sequences have higher probabilities e.g:

- $p(\text{"I like cats"}) > p(\text{"I table cats"})$
- $p(\text{"I like cats"}) > p(\text{"like I cats"})$

**Auto-regressive sequence modelling**

$$p_\theta(w_0) \cdot p_\theta(w_1 | w_0)$$

$p_\theta$ is parametrized by a neural network.

# Language Models

Assign a probability to a sequence of words, such that plausible sequences have higher probabilities e.g:

- $p(\text{"I like cats"}) > p(\text{"I table cats"})$
- $p(\text{"I like cats"}) > p(\text{"like I cats"})$

**Auto-regressive sequence modelling**

$$p_\theta(w_0) \cdot p_\theta(w_1|w_0) \cdot \ldots \cdot p_\theta(w_n|w_{n-1}, w_{n-2}, \ldots, w_0)$$

$p_\theta$ is parametrized by a neural network.

# Language Models

Assign a probability to a sequence of words, such that plausible sequences have higher probabilities e.g:

- $p(\text{"I like cats"}) > p(\text{"I table cats"})$
- $p(\text{"I like cats"}) > p(\text{"like I cats"})$

**Auto-regressive sequence modelling**

$$p_\theta(w_0) \cdot p_\theta(w_1|w_0) \cdot \ldots \cdot p_\theta(w_n|w_{n-1}, w_{n-2}, \ldots, w_0)$$

$p_\theta$ is parametrized by a neural network.

The internal representation of the model can better capture the meaning of a sequence than a simple Bag-of-Words.

# Conditional Language Models

NLP problems expressed as **Conditional Language Models**:

**Translation:** $p(Target|Source)$

- *Source*: "J'aime les chats"
- *Target*: "I like cats"

# Conditional Language Models

NLP problems expressed as **Conditional Language Models**:

**Translation:** $p(Target|Source)$

- *Source*: "J'aime les chats"
- *Target*: "I like cats"

Model the output word by word:

$p_\theta(w_0|Source)$

# Conditional Language Models

NLP problems expressed as **Conditional Language Models**:

**Translation:** $p(Target|Source)$

- *Source*: "J'aime les chats"
- *Target*: "I like cats"

Model the output word by word:

$$p_\theta(w_0|Source) \cdot p_\theta(w_1|w_0, Source) \cdot \ldots$$

# Conditional Language Models

**Question Answering / Dialogue:**

$p(Answer|Question, Context)$

- *Context*:
    - "John puts two glasses on the table."
    - "Bob adds two more glasses."
    - "Bob leaves the kitchen to play baseball in the garden."
- *Question*: "How many glasses are there?"
- *Answer*: "There are four glasses."

# Conditional Language Models

**Question Answering / Dialogue:**

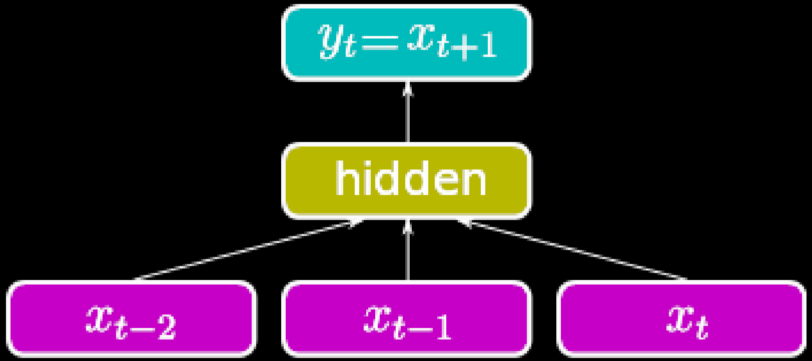
$p(Answer|Question, Context)$

- *Context*:
  - "John puts two glasses on the table."
  - "Bob adds two more glasses."
  - "Bob leaves the kitchen to play baseball in the garden."
- *Question*: "How many glasses are there?"
- *Answer*: "There are four glasses."

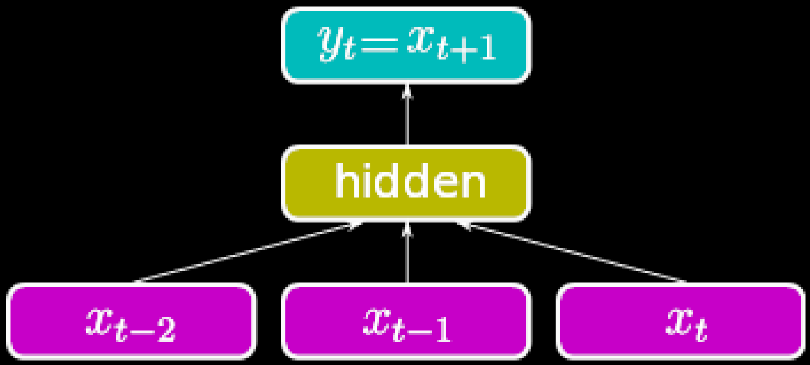
**Image Captionning:** $p(Caption|Image)$

- Instead of raw image, instead represent using activation of the penultimate layer of a CNN

# Simple Language Model



# Simple Language Model

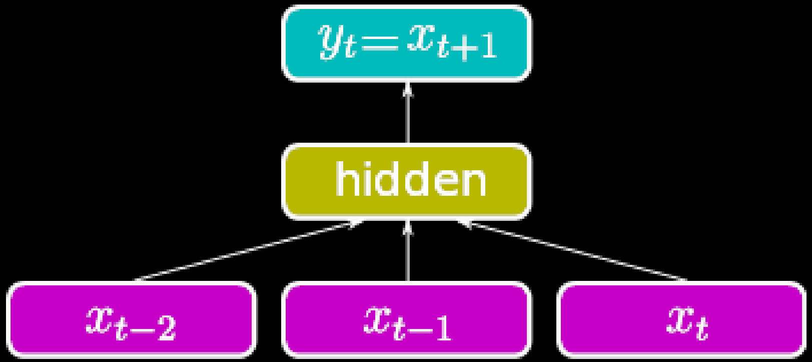



Fixed context size

Different modelling choices:

- **Average embeddings**: (same as CBoW) no sequence information, ie permutation invariant

# Simple Language Model

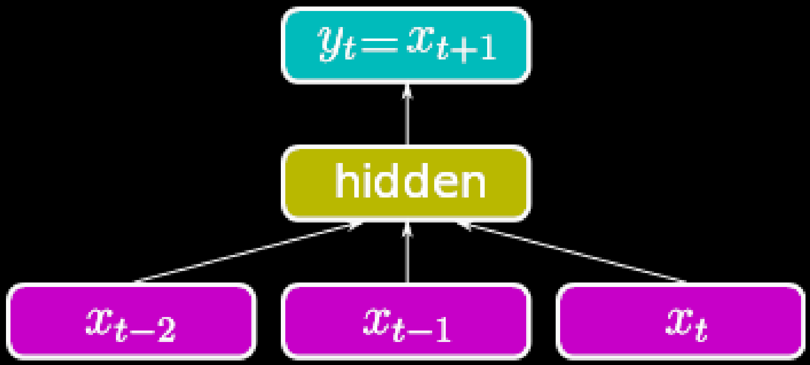



Fixed context size

Different modelling choices:

- **Average embeddings**: (same as CBoW) no sequence information, ie permutation invariant
- **Concatenate embeddings**: introduces many parameters

# Simple Language Model

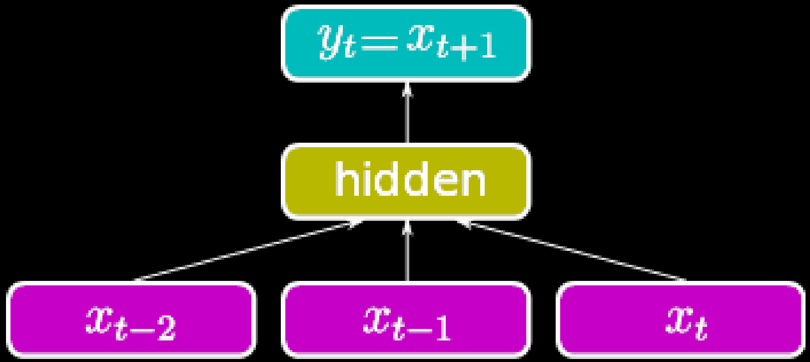



Fixed context size

Different modelling choices:

- **Average embeddings**: (same as CBoW) no sequence information, ie permutation invariant
- **Concatenate embeddings**: introduces many parameters
- **1D convolution**: larger contexts and limit number of parameters

# Simple Language Model

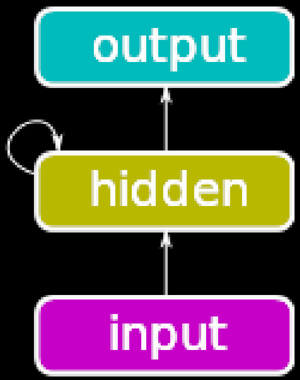



Fixed context size

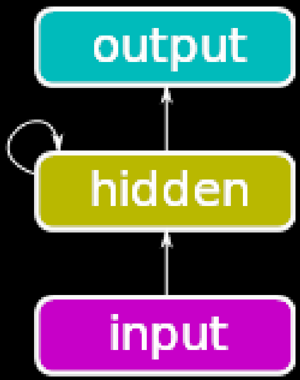
Different modelling choices:

- **Average embeddings**: (same as CBoW) no sequence information, ie permutation invariant
- **Concatenate embeddings**: introduces many parameters
- **1D convolution**: larger contexts and limit number of parameters
- Still does not take well into account varying sequence sizes and sequence dependencies

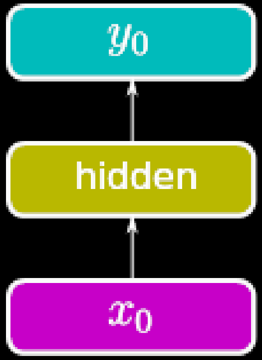
# Recurrent Neural Network



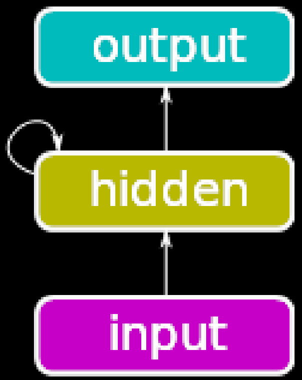
# Recurrent Neural Network



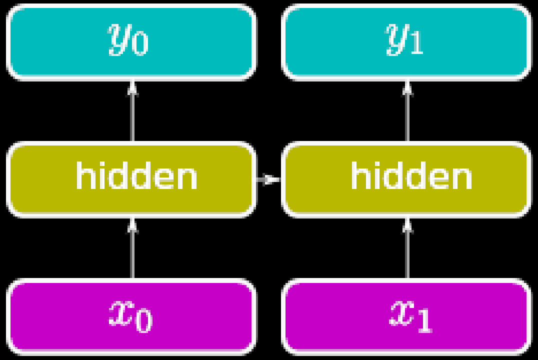

Unroll over a sequence $(x_0, x_1, x_2)$:



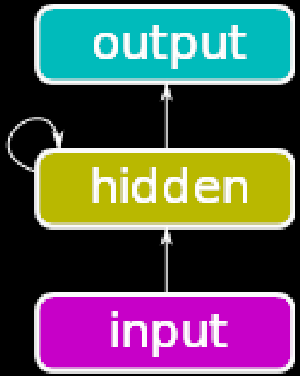
# Recurrent Neural Network



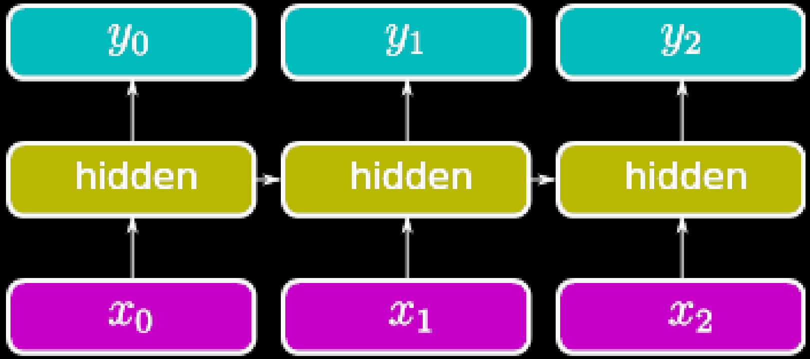

Unroll over a sequence $(x_0, x_1, x_2)$:



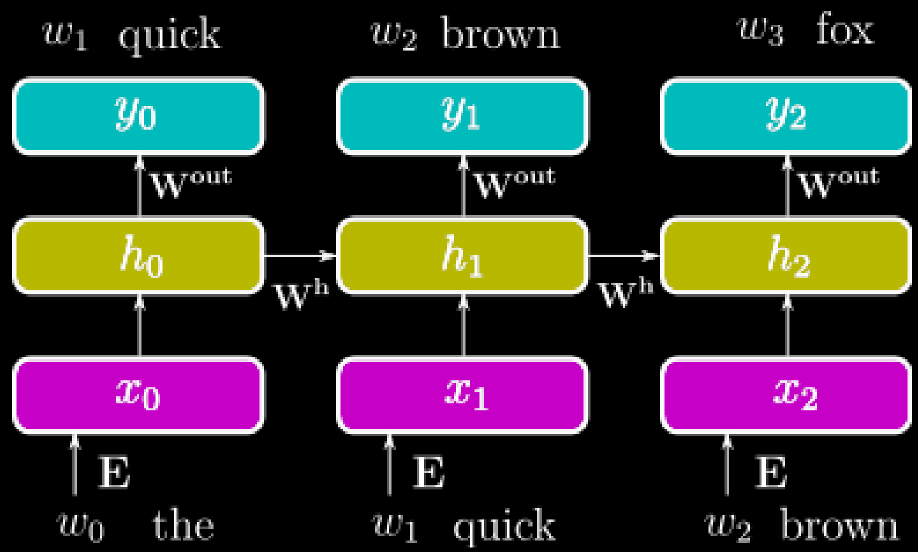
# Recurrent Neural Network




Unroll over a sequence $(x_0, x_1, x_2)$:

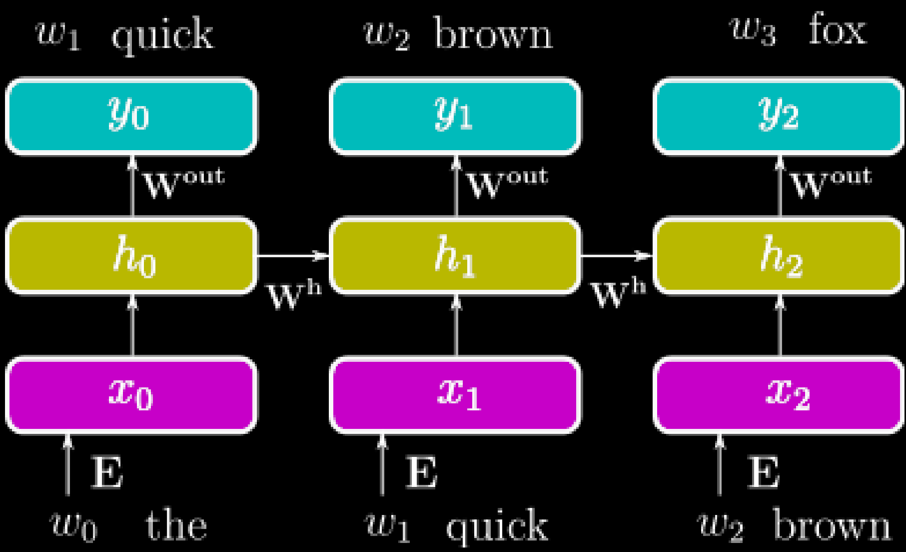


# Language Modelling




**input** $(w_0, w_1, \ldots, w_t)$ sequence of words ( 1-hot encoded )
**output** $(w_1, w_2, \ldots, w_{t+1})$ shifted sequence of words ( 1-hot encoded )

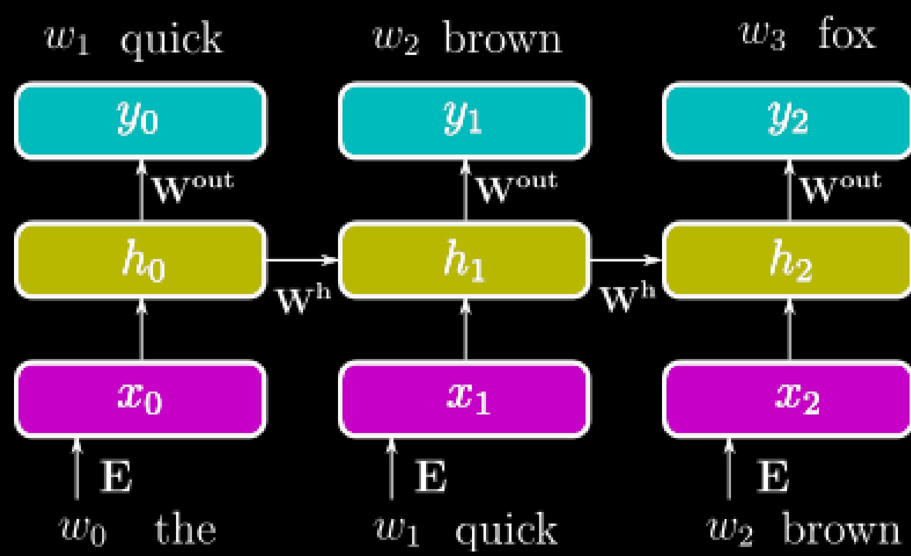
# Language Modelling




$$x_t = \text{Emb}(w_t) = \mathbf{E}w_t$$

input projection **H**

# Language Modelling



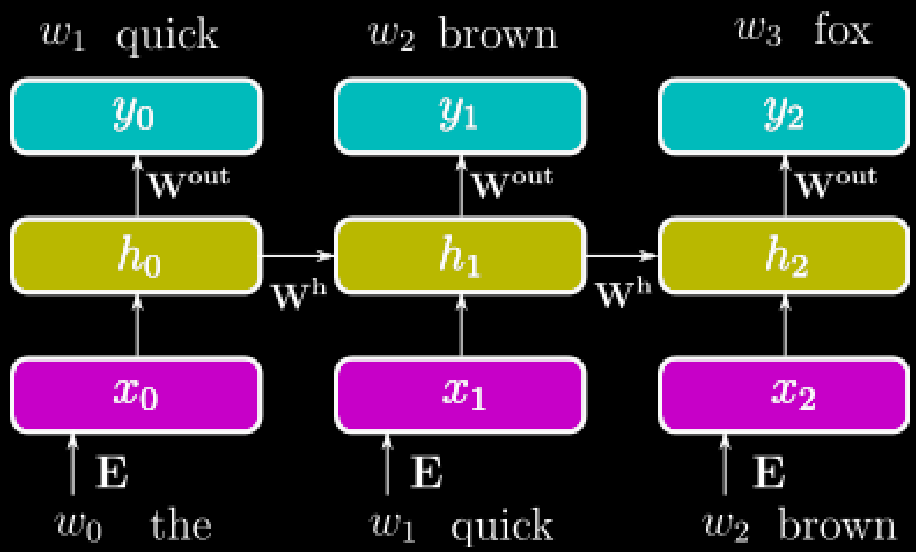

$$x_t = \text{Emb}(w_t) = \mathbf{E}w_t$$

input projection **H**

$$h_t = g(\mathbf{W^h} h_{t-1} + x_t + b^h)$$

recurrent connection **H**

# Language Modelling




$$x_t = \text{Emb}(w_t) = \mathbf{E}w_t$$

input projection H

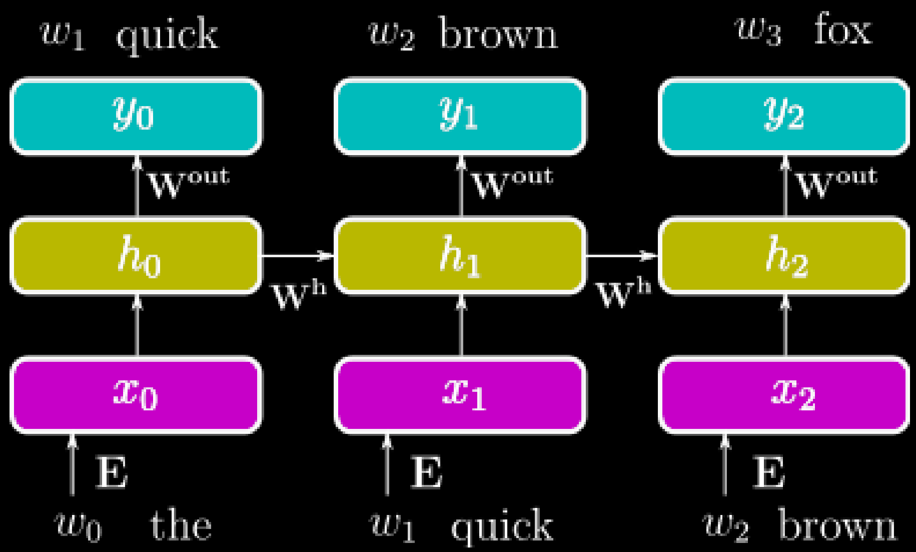
$$h_t = g(\mathbf{W^h} h_{t-1} + x_t + b^h)$$

recurrent connection H

$$y = \text{softmax}(\mathbf{W^o} h_t + b^o)$$

output projection K = |V|

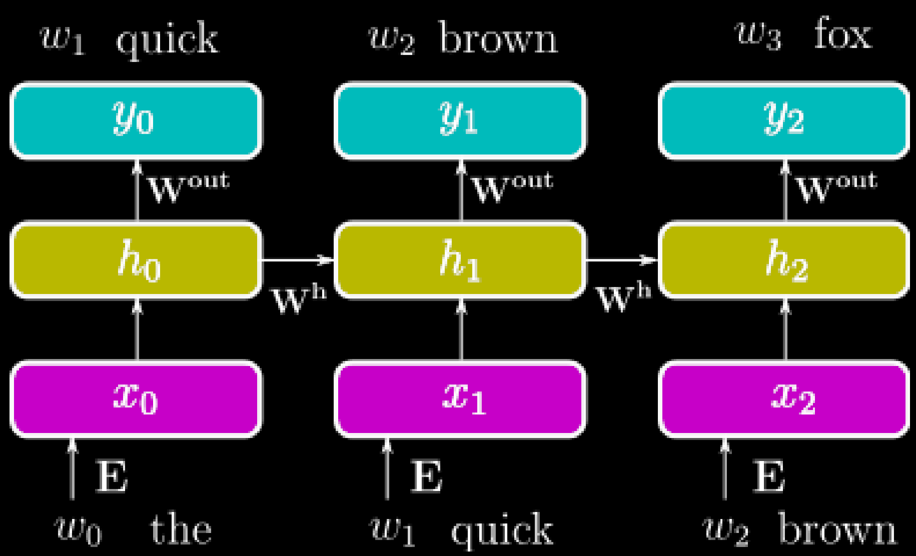
# Recurrent Neural Network




Input embedding $\mathbf{E}$

|V| x H

# Recurrent Neural Network



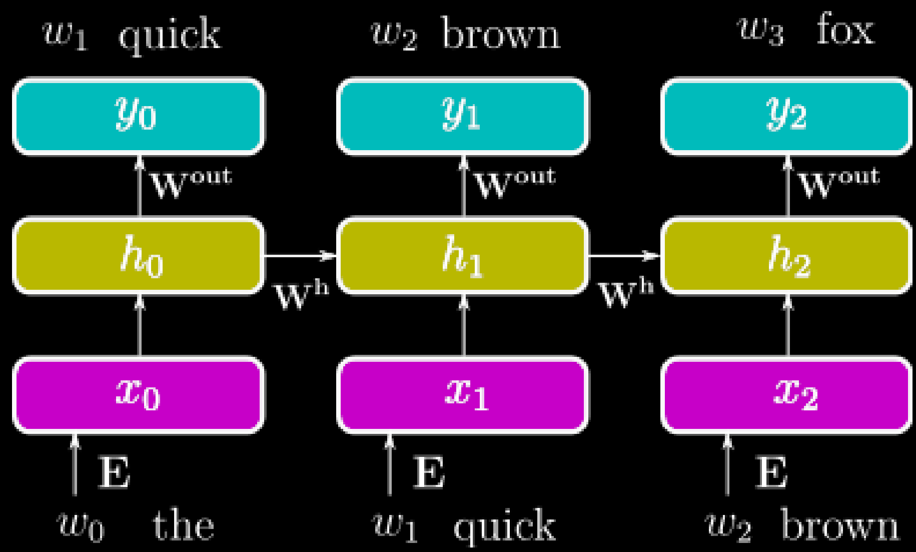

Input embedding $\mathbf{E}$

|V| x H

Recurrent weights $\mathbf{W^h}$

H x H

# Recurrent Neural Network




Input embedding $\mathbf{E}$

|V| x H

Recurrent weights $\mathbf{W}^{\mathbf{h}}$

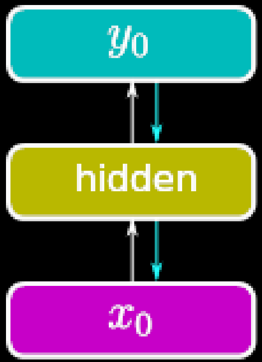
H x H

Output weights $\mathbf{W}^{\mathbf{out}}$

H x K = H x |V|

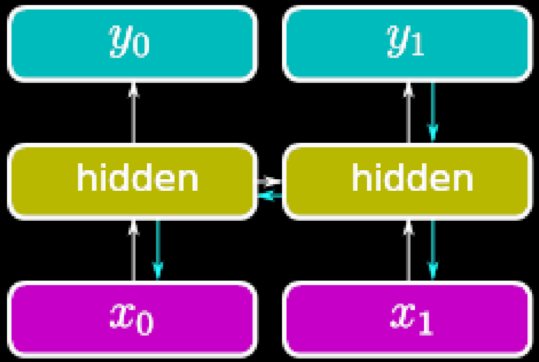
# Backpropagation through time

Similar as standard backpropagation on unrolled network



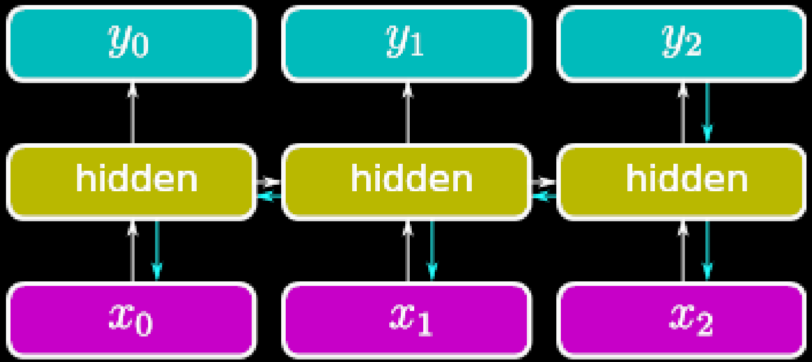
# Backpropagation through time

Similar as standard backpropagation on unrolled network



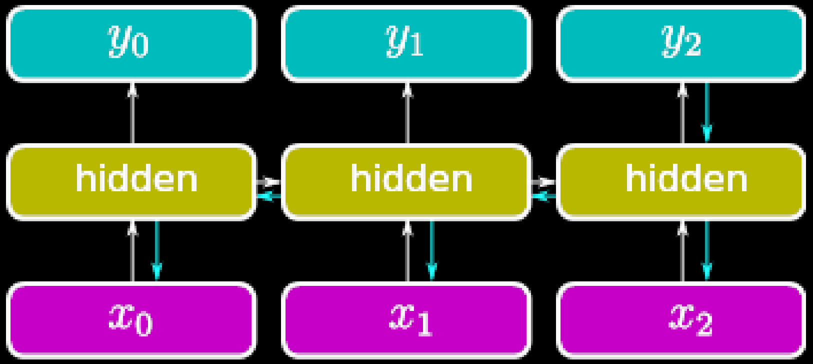
# Backpropagation through time

Similar as standard backpropagation on unrolled network



# Backpropagation through time

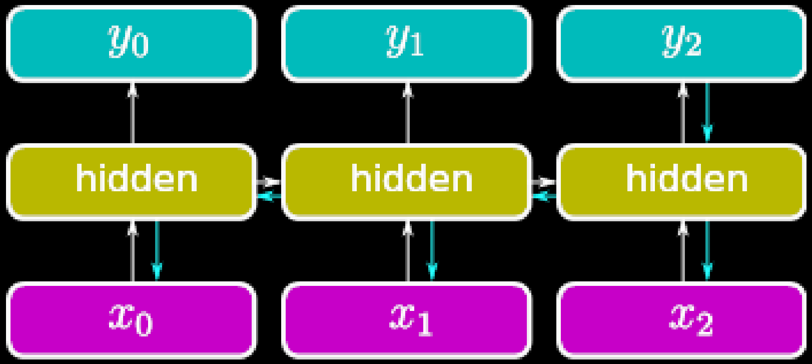
Similar as standard backpropagation on unrolled network




- Similar as training **very deep networks** with tied parameters
- Example between $x_0$ and $y_2$: $W^h$ is used twice

# Backpropagation through time

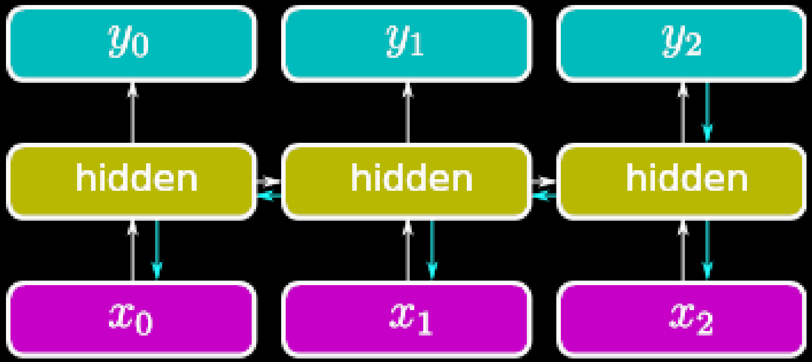
Similar as standard backpropagation on unrolled network




- Similar as training **very deep networks** with tied parameters
- Example between $x_0$ and $y_2$: $W^h$ is used twice
- Usually truncate the backprop after $T$ timesteps

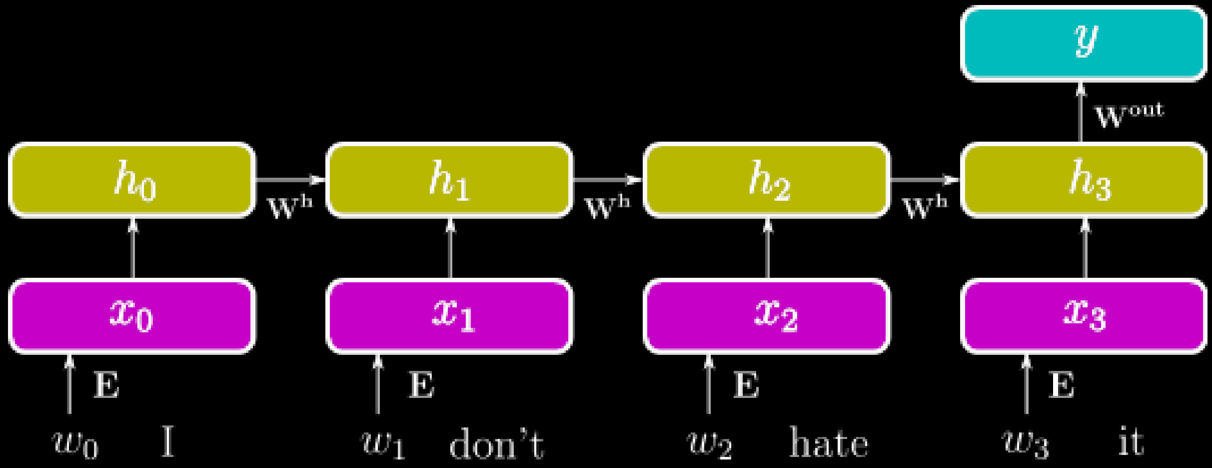
# Backpropagation through time

Similar as standard backpropagation on unrolled network



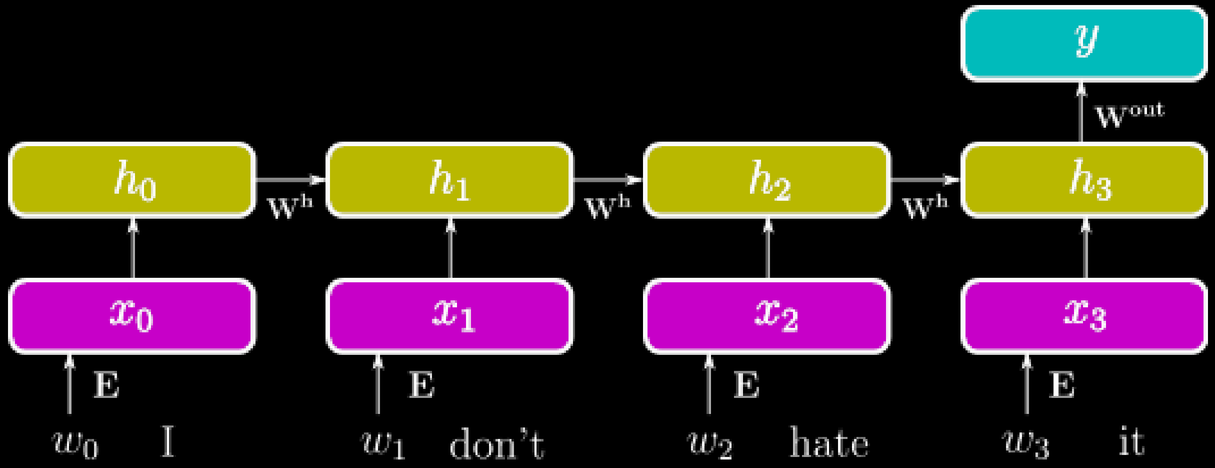

- Similar as training **very deep networks** with tied parameters
- Example between $x_0$ and $y_2$: $W^h$ is used twice
- Usually truncate the backprop after $T$ timesteps
- Difficulties to train long-term dependencies

# Other uses: Sentiment Analysis



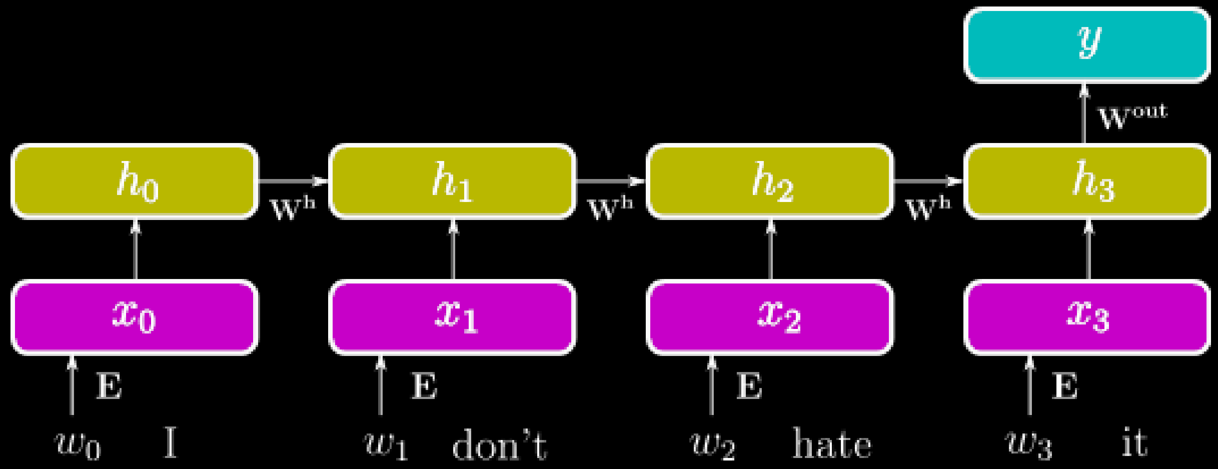

- Output is sentiment (1 for positive, 0 for negative)

# Other uses: Sentiment Analysis



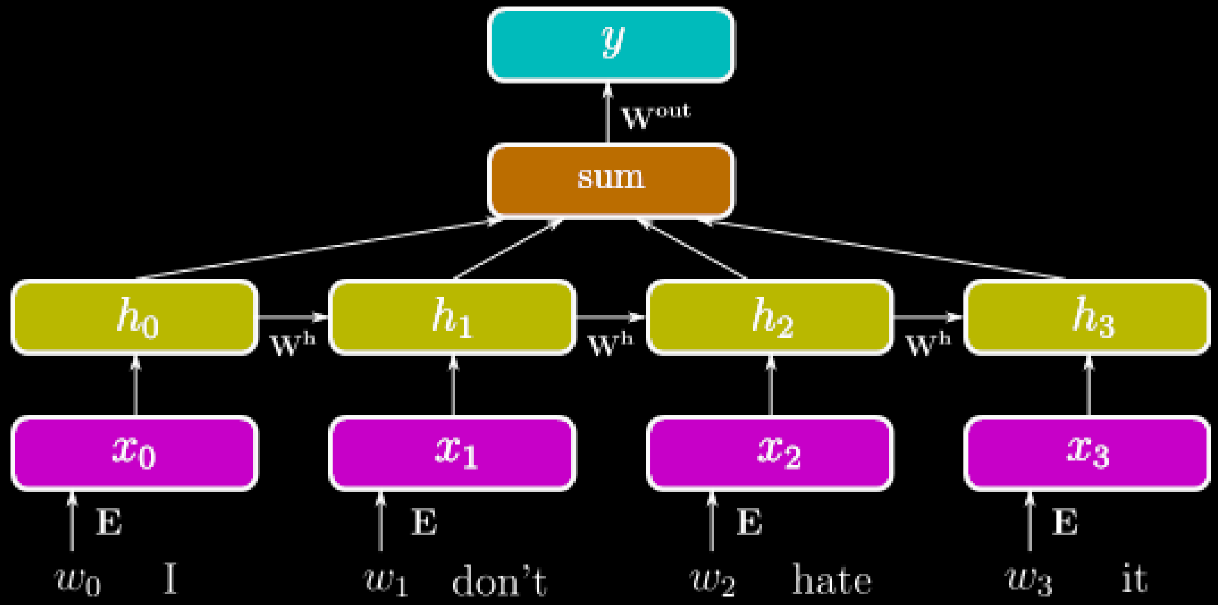

- Output is sentiment (1 for positive, 0 for negative)
- Very dependent on words order

# Other uses: Sentiment Analysis




- Output is sentiment (1 for positive, 0 for negative)
- Very dependent on words order
- Very flexible network architectures

# Other uses: Sentiment analysis




- Output is sentiment (1 for positive, 0 for negative)
- Very dependent on words order
- Very flexible network architectures

# Vanishing / Exploding Gradients

Passing through $t$ time-steps, the resulting gradient is the **product** of many gradients and activations.

# Vanishing / Exploding Gradients

Passing through $t$ time-steps, the resulting gradient is the **product** of many gradients and activations.

- Gradient messages close to $0$ can shrink be $0$
- Gradient messages larger than $1$ can explode

# Vanishing / Exploding Gradients

Passing through $t$ time-steps, the resulting gradient is the **product** of many gradients and activations.

- Gradient messages close to $0$ can shrink be $0$
- Gradient messages larger than $1$ can explode
- **LSTM / GRU** mitigate that in RNNs
- **Additive path** between $c_t$ and $c_{t-1}$

# Vanishing / Exploding Gradients

Passing through $t$ time-steps, the resulting gradient is the **product** of many gradients and activations.

- Gradient messages close to $0$ can shrink be $0$
- Gradient messages larger than $1$ can explode
- **LSTM / GRU** mitigate that in RNNs
- **Additive path** between $c_t$ and $c_{t-1}$
- **Gradient clipping** prevents gradient explosion
- Well chosen **activation function** is critical (tanh)

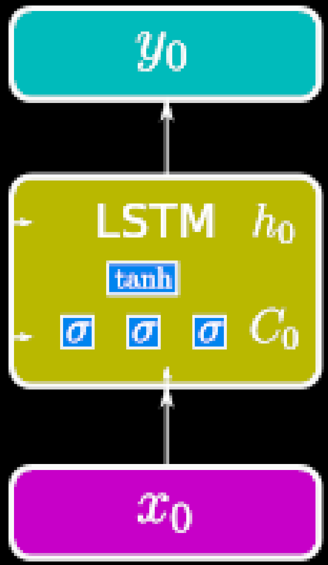
# Vanishing / Exploding Gradients

Passing through $t$ time-steps, the resulting gradient is the **product** of many gradients and activations.

- Gradient messages close to $0$ can shrink be $0$
- Gradient messages larger than $1$ can explode
- **LSTM / GRU** mitigate that in RNNs
- **Additive path** between $c_t$ and $c_{t-1}$
- **Gradient clipping** prevents gradient explosion
- Well chosen **activation function** is critical (tanh)

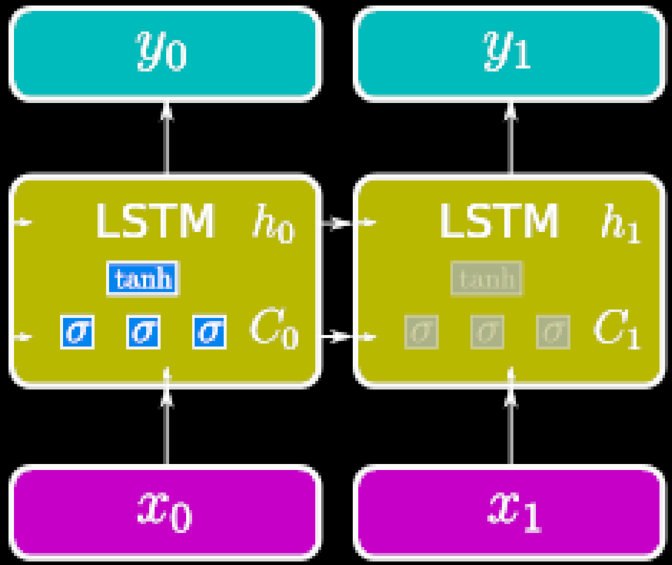
**Skip connections** in ResNet also alleviate a similar optimization problem.

# LSTM



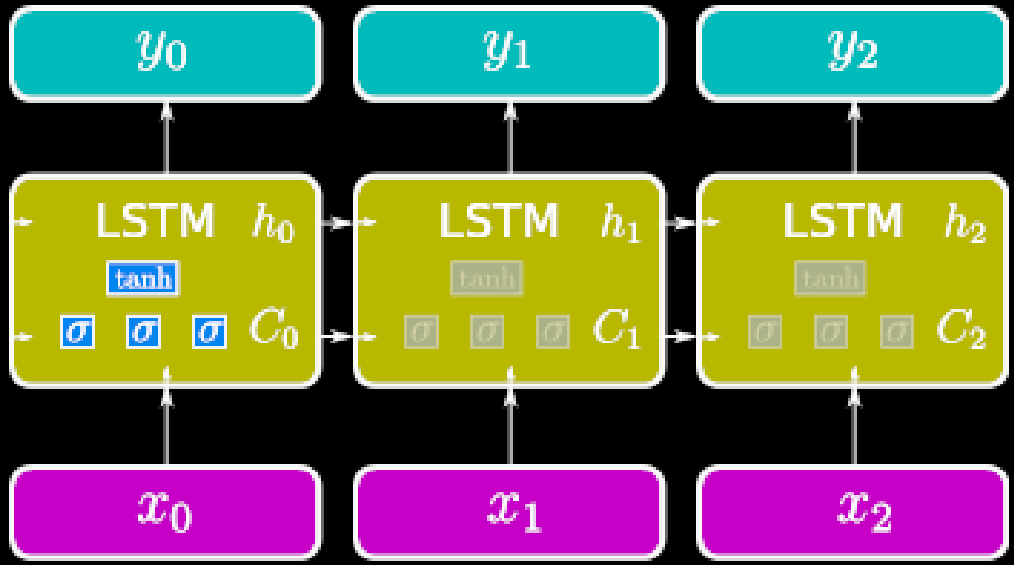

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 1997

# LSTM



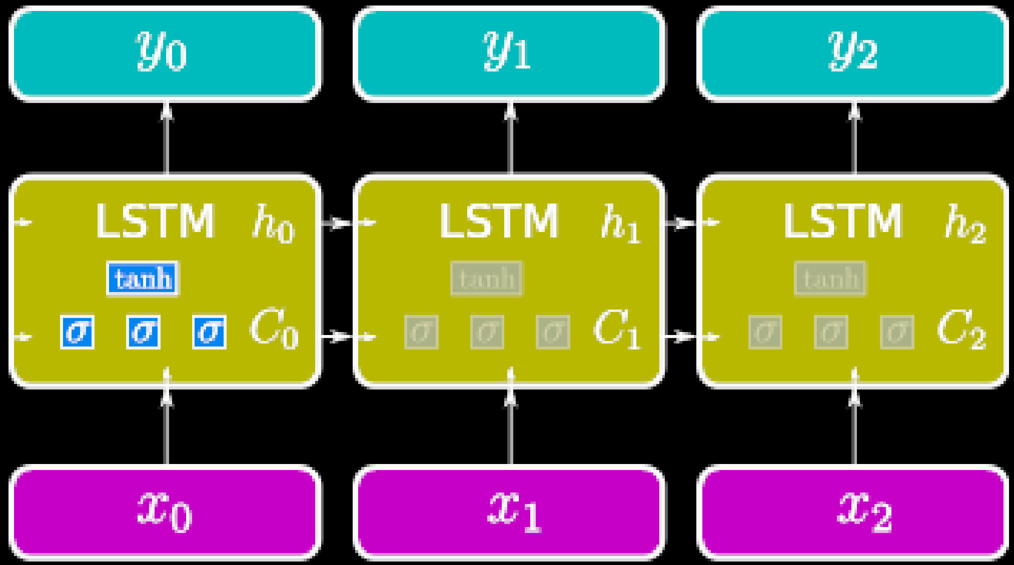

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 1997

# LSTM




Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 1997

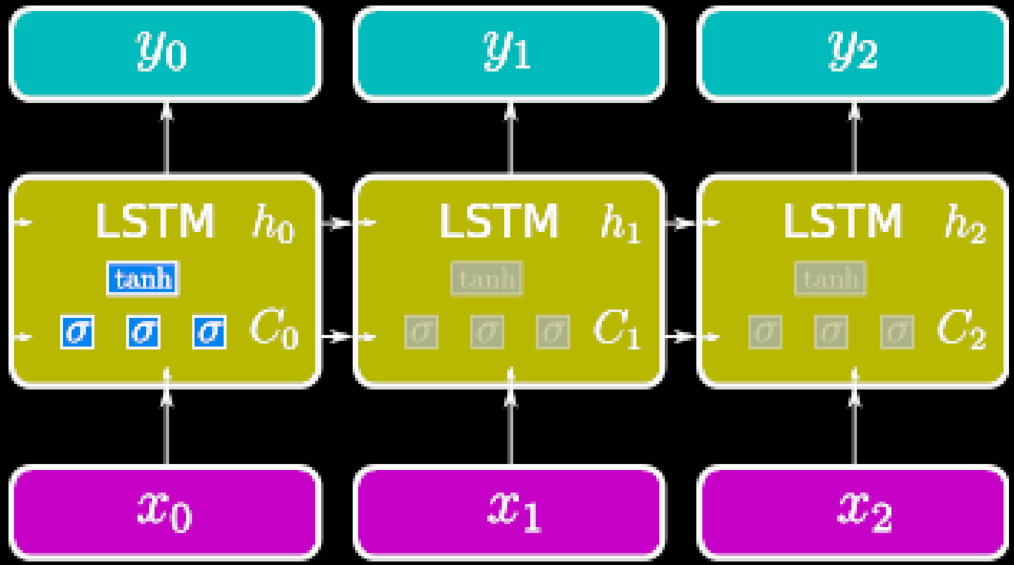
# LSTM




- 4 times more parameters than RNN

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 1997

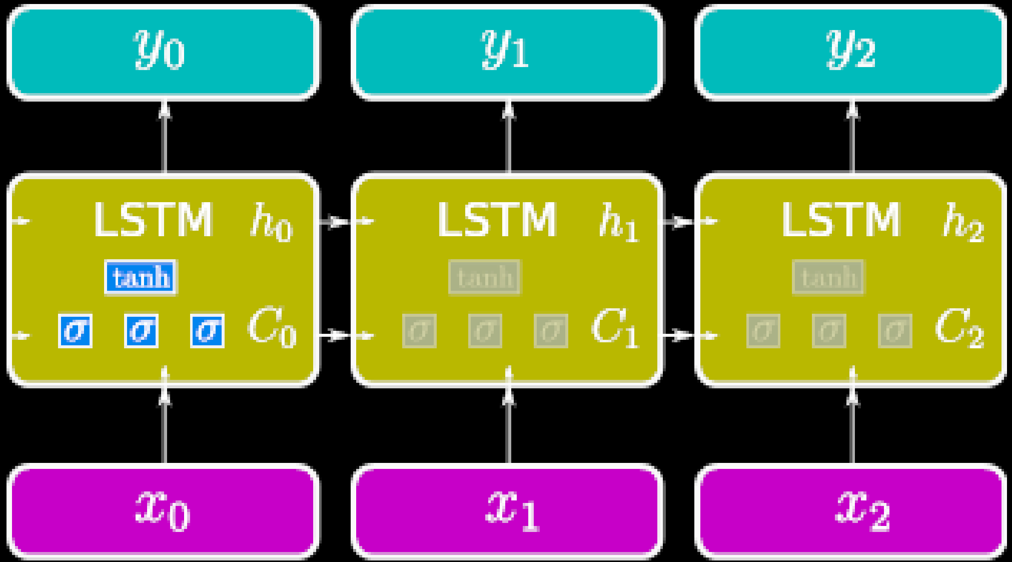
# LSTM




- 4 times more parameters than RNN
- Mitigates **vanishing gradient** problem through **gating**

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 1997

# LSTM




- 4 times more parameters than RNN
- Mitigates **vanishing gradient** problem through **gating**
- Widely used and (up until recently) SOTA in many sequence learning problems

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 1997

$$\mathbf{u} = \sigma(\mathbf{W^u} \cdot h_{t-1} + \mathbf{I^u} \cdot x_t + b^u)$$

Update gate **H**

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 1997

$$\mathbf{u} = \sigma(\mathbf{W^u} \cdot h_{t-1} + \mathbf{I^u} \cdot x_t + b^u)$$

Update gate **H**

$$\mathbf{f} = \sigma(\mathbf{W^f} \cdot h_{t-1} + \mathbf{I^f} \cdot x_t + b^f)$$

Forget gate **H**

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 1997

$$\mathbf{u} = \sigma(\mathbf{W^u} \cdot h_{t-1} + \mathbf{I^u} \cdot x_t + b^u)$$

Update gate **H**

$$\mathbf{f} = \sigma(\mathbf{W^f} \cdot h_{t-1} + \mathbf{I^f} \cdot x_t + b^f)$$

Forget gate **H**

$$\tilde{\mathbf{c}}_\mathbf{t} = \tanh(\mathbf{W^c} \cdot h_{t-1} + \mathbf{I^c} \cdot x_t + b^c)$$

Cell candidate **H**

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 1997

$$\mathbf{u} = \sigma(\mathbf{W^u} \cdot h_{t-1} + \mathbf{I^u} \cdot x_t + b^u)$$

Update gate **H**

$$\mathbf{f} = \sigma(\mathbf{W^f} \cdot h_{t-1} + \mathbf{I^f} \cdot x_t + b^f)$$

Forget gate **H**

$$\mathbf{\tilde{c_t}} = \tanh(\mathbf{W^c} \cdot h_{t-1} + \mathbf{I^c} \cdot x_t + b^c)$$

Cell candidate **H**

$$\mathbf{c_t} = \mathbf{f} \odot \mathbf{c_{t-1}} + \mathbf{u} \odot \mathbf{\tilde{c_t}}$$

Cell output **H**

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 1997

$$\mathbf{u} = \sigma(\mathbf{W^u} \cdot h_{t-1} + \mathbf{I^u} \cdot x_t + b^u)$$

Update gate **H**

$$\mathbf{f} = \sigma(\mathbf{W^f} \cdot h_{t-1} + \mathbf{I^f} \cdot x_t + b^f)$$

Forget gate **H**

$$\mathbf{\tilde{c_t}} = \tanh(\mathbf{W^c} \cdot h_{t-1} + \mathbf{I^c} \cdot x_t + b^c)$$

Cell candidate **H**

$$\mathbf{c_t} = \mathbf{f} \odot \mathbf{c_{t-1}} + \mathbf{u} \odot \mathbf{\tilde{c_t}}$$

Cell output **H**

$$\mathbf{o} = \sigma(\mathbf{W^o} \cdot h_{t-1} + \mathbf{I^o} \cdot x_t + b^o)$$

Output gate **H**

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 1997

$$\mathbf{u} = \sigma(\mathbf{W^u} \cdot h_{t-1} + \mathbf{I^u} \cdot x_t + b^u)$$

Update gate **H**

$$\mathbf{f} = \sigma(\mathbf{W^f} \cdot h_{t-1} + \mathbf{I^f} \cdot x_t + b^f)$$

Forget gate **H**

$$\mathbf{\tilde{c}_t} = \tanh(\mathbf{W^c} \cdot h_{t-1} + \mathbf{I^c} \cdot x_t + b^c)$$

Cell candidate **H**

$$\mathbf{c_t} = \mathbf{f} \odot \mathbf{c_{t-1}} + \mathbf{u} \odot \mathbf{\tilde{c}_t}$$

Cell output **H**

$$\mathbf{o} = \sigma(\mathbf{W^o} \cdot h_{t-1} + \mathbf{I^o} \cdot x_t + b^o)$$

Output gate **H**

$$\mathbf{h_t} = \mathbf{o} \odot \tanh(\mathbf{c_t})$$

Hidden output **H**

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 1997

$$\mathbf{u} = \sigma(\mathbf{W^u} \cdot h_{t-1} + \mathbf{I^u} \cdot x_t + b^u)$$

Update gate **H**

$$\mathbf{f} = \sigma(\mathbf{W^f} \cdot h_{t-1} + \mathbf{I^f} \cdot x_t + b^f)$$

Forget gate **H**

$$\tilde{\mathbf{c_t}} = \tanh(\mathbf{W^c} \cdot h_{t-1} + \mathbf{I^c} \cdot x_t + b^c)$$

Cell candidate **H**

$$\mathbf{c_t} = \mathbf{f} \odot \mathbf{c_{t-1}} + \mathbf{u} \odot \tilde{\mathbf{c_t}}$$

Cell output **H**

$$\mathbf{o} = \sigma(\mathbf{W^o} \cdot h_{t-1} + \mathbf{I^o} \cdot x_t + b^o)$$

Output gate **H**

$$\mathbf{h_t} = \mathbf{o} \odot \tanh(\mathbf{c_t})$$

Hidden output **H**

$$y = \mathrm{softmax}(\mathbf{W} \cdot h_t + b)$$

Output **K**

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 1997

$$\mathbf{u} = \sigma(\mathbf{W^u} \cdot h_{t-1} + \mathbf{I^u} \cdot x_t + b^u)$$

Update gate **H**

$$\mathbf{f} = \sigma(\mathbf{W^f} \cdot h_{t-1} + \mathbf{I^f} \cdot x_t + b^f)$$

Forget gate **H**

$$\mathbf{\tilde{c_t}} = \tanh(\mathbf{W^c} \cdot h_{t-1} + \mathbf{I^c} \cdot x_t + b^c)$$

Cell candidate **H**

$$\mathbf{c_t} = \mathbf{f} \odot \mathbf{c_{t-1}} + \mathbf{u} \odot \mathbf{\tilde{c_t}}$$

Cell output **H**

$$\mathbf{o} = \sigma(\mathbf{W^o} \cdot h_{t-1} + \mathbf{I^o} \cdot x_t + b^o)$$

Output gate **H**

$$\mathbf{h_t} = \mathbf{o} \odot \tanh(\mathbf{c_t})$$

Hidden output **H**

$$y = \mathrm{softmax}(\mathbf{W} \cdot h_t + b)$$

Output **K**

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 1997

# GRU

Gated Recurrent Unit: similar idea as LSTM

- less parameters, as there is one gate less
- no "cell", only hidden vector $h_t$ is passed to next unit

Chung, Junyoung, et al. "Gated Feedback Recurrent Neural Networks." ICML 2015

# GRU

Gated Recurrent Unit: similar idea as LSTM

- less parameters, as there is one gate less
- no "cell", only hidden vector $h_t$ is passed to next unit

In practice

- more recent, people tend to use LSTM more
- no systematic difference between the two

Chung, Junyoung, et al. "Gated Feedback Recurrent Neural Networks." ICML 2015

# GPT-2: Current SOTA generation

–

```
I know that some people might be opposed to wearing
sneakers in a turtleneck, but I wanted to be true to myself,
so I went with a slim fitting turtleneck in a color more
similar to my favorite color of the day. Overall, it’s a
classic turtleneck, with nothing too flashy. I did swap my
shoes for these ASICS Tiger Killshots and I wish I hadn’t
did that. The shoe itself is very comfortable, however,
I found that wearing them in these sneakers made my
feet look a bit larger. I’d wear them with denim, chinos,
whatever, but would never recommend wearing them
alone. There are just too many variables involved.
```

# GPT-2: Current SOTA generation

–

```
Captain James J. Dall was born in Virginia in 1829 during
the Revolution and joined his father in the Army in 1836.
He served with the 2d Virginia Infantry until 1845, when
he went to the Western Reserve Regiment; and, on his
discharge, he reenlisted with Company I, 25th New York
Volunteers (which in 1845 numbered about 4,500 men).
One of his brothers, James, served with his company
in the same capacity in the same brigade. While the
regiment remained in Connecticut through 1846, Captain
Dall served with Company I at various times before
his discharge. A veteran of the Mexican War, he had
served two years on the staff of Brigadier General John J.
Sullivan at San Antonio. During the Peninsula Campaign
of 1863, he commanded a company of twenty-three and
was in charge of the battery of fifty-four heavy mortars
and the gunboat Pinta of the Battery, a gunboat and
several other boats. Captain Dall was on active duty with
Company I in the fall of 1865. Tw
```

# GPT-2: Current SOTA generation

–

```
Methinks I see her in her blissful dreams:
Or, fancy-like, in some mirage she lies,
Majestic yet majestic, and of seems
The image of the unconquerable skies.

Methinks I see her in her blissful dreams:
—Or, fancy-like, in some majestic cell,
Where lordly seraphs strew their balmy dreams
On the still night, or in their golden shell.

There, in the calm of some Platonic dream,
Sits she, and views the unclouded moon arise
Like a fair lady full of realms divine;

And, all at once, a stony face and bright
Glittering in moonlight, like the noon-tints of a night.
```

# Thanks to

Charles Ollion and Olivier Grisel, Paris-Saclay, for the slides

Chris Holmes and Gil McVean, as always