

Natural Language Processing with Deep Learning

Charles Ollion - Olivier Grisel

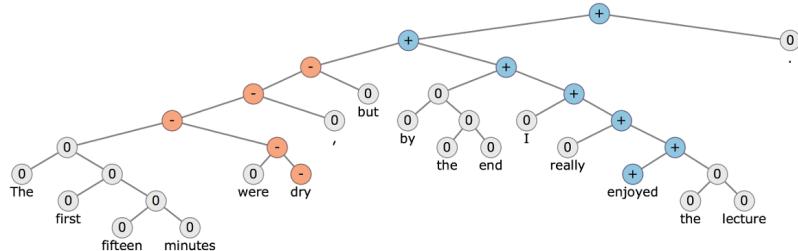


MASTER
DATASCIENCE
UNIVERSITÉ PARIS-SACLAY

Natural Language Processing



[Google Translate System - 2016]



[Socher 2015]

Natural Language Processing

- Sentence/Document level Classification (topic, sentiment)
- Topic modeling (LDA, ...)

Natural Language Processing

- Sentence/Document level Classification (topic, sentiment)
- Topic modeling (LDA, ...)
- Translation

Natural Language Processing

- Sentence/Document level Classification (topic, sentiment)
- Topic modeling (LDA, ...)
- Translation
- Chatbots / dialogue systems / assistants (Alexa, ...)

Natural Language Processing

- Sentence/Document level Classification (topic, sentiment)
- Topic modeling (LDA, ...)
- Translation
- Chatbots / dialogue systems / assistants (Alexa, ...)
- Summarization

Recommended reading

A Primer on Neural Network Models for Natural Language Processing by Yoav Goldberg

<http://u.cs.biu.ac.il/~yogo/nlp.pdf>

Recommended reading

A Primer on Neural Network Models for Natural Language Processing by Yoav Goldberg

<http://u.cs.biu.ac.il/~yogo/nlp.pdf>

Useful open source projects

gensim spaCy



Outline

Classification and word representation

Outline

Classification and word representation

Word2Vec

Outline

Classification and word representation

Word2Vec

Language Modelling

Outline

Classification and word representation

Word2Vec

Language Modelling

Recurrent neural networks

Word Representation and Word2Vec

Word representation

Words are indexed and represented as 1-hot vectors

Word representation

Words are indexed and represented as 1-hot vectors

Large Vocabulary of possible words $|V|$

Word representation

Words are indexed and represented as 1-hot vectors

Large Vocabulary of possible words $|V|$

Use of **Embeddings** as inputs in all Deep NLP tasks

Word representation

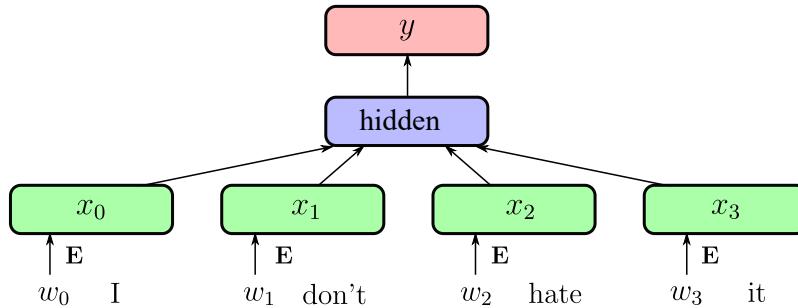
Words are indexed and represented as 1-hot vectors

Large Vocabulary of possible words $|V|$

Use of **Embeddings** as inputs in all Deep NLP tasks

Word embeddings usually have dimensions 50, 100, 200, 300

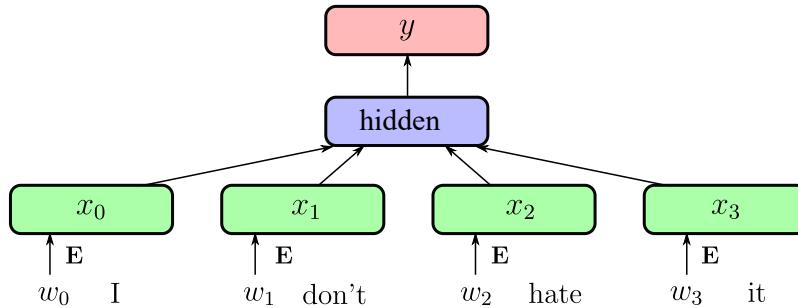
Supervised Text Classification



Joulin, Armand, et al. "Bag of tricks for efficient text classification." FAIR 2016

18 / 112

Supervised Text Classification



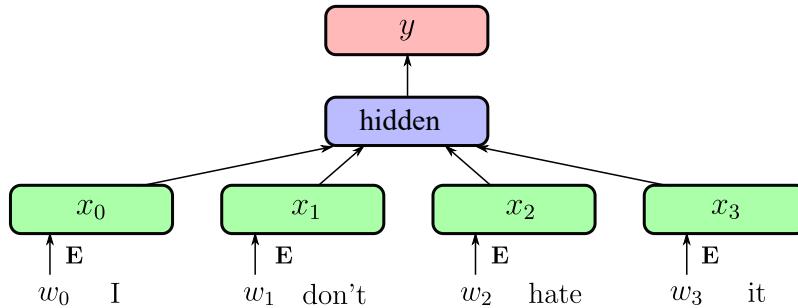
E embedding (linear projection)

|V| \times H

Joulin, Armand, et al. "Bag of tricks for efficient text classification." FAIR 2016

19 / 112

Supervised Text Classification



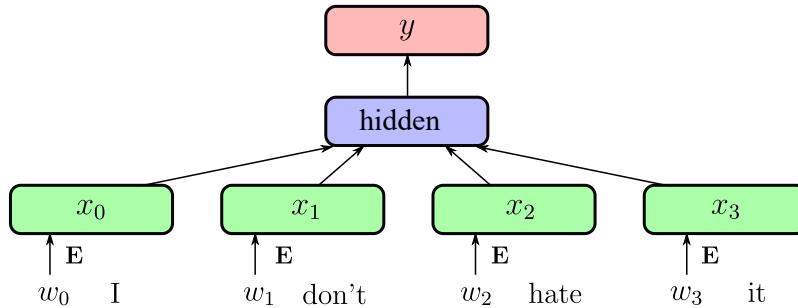
E embedding (linear projection)

$|V| \times H$

Embeddings are averaged

hidden activation size: **H**

Supervised Text Classification



E embedding (linear projection)

$|V| \times H$

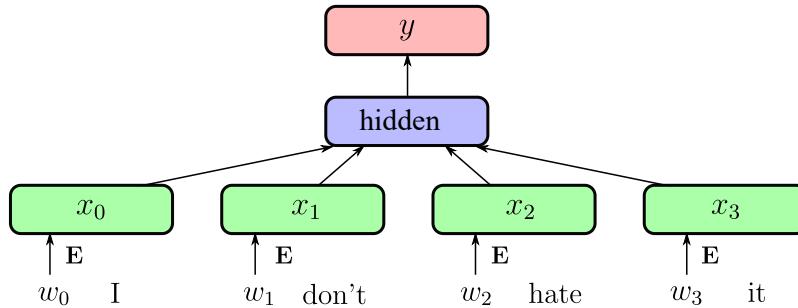
Embeddings are averaged

hidden activation size: H

Dense output connection \mathbf{W}, \mathbf{b}

$H \times K$

Supervised Text Classification



E embedding (linear projection)

$|V| \times H$

Embeddings are averaged

hidden activation size: H

Dense output connection \mathbf{W}, \mathbf{b}

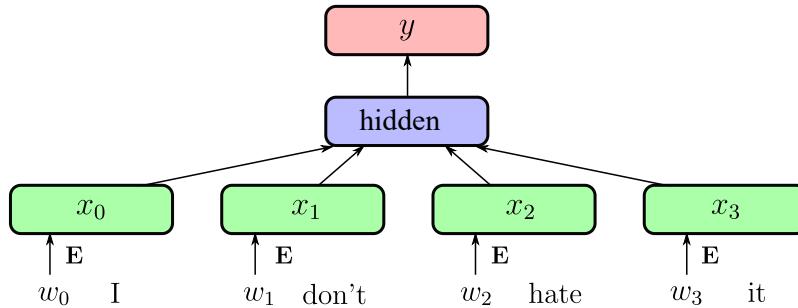
$H \times K$

Softmax and cross-entropy loss

Joulin, Armand, et al. "Bag of tricks for efficient text classification." FAIR 2016

22 / 112

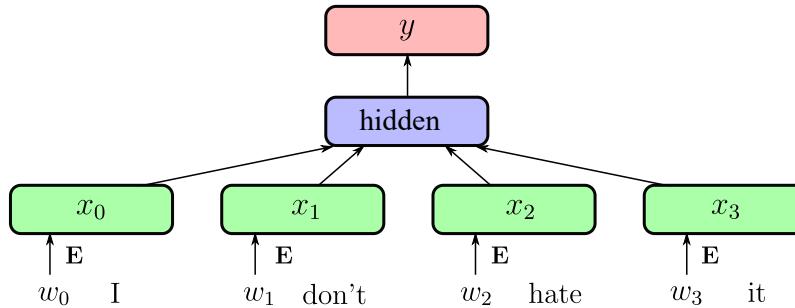
Supervised Text Classification



- Very efficient (**speed and accuracy**) on large datasets

Joulin, Armand, et al. "Bag of tricks for efficient text classification." FAIR 2016

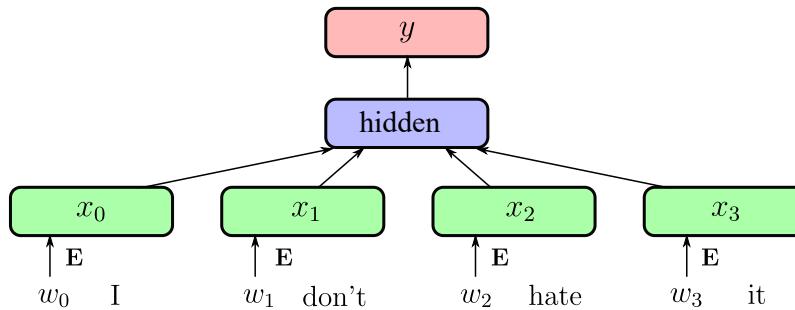
Supervised Text Classification



- Very efficient (**speed and accuracy**) on large datasets
- State-of-the-art (or close to) on several classification, when adding **bigrams/trigrams**

Joulin, Armand, et al. "Bag of tricks for efficient text classification." FAIR 2016

Supervised Text Classification



- Very efficient (**speed and accuracy**) on large datasets
- State-of-the-art (or close to) on several classification, when adding **bigrams/trigrams**
- Little gains from depth

Joulin, Armand, et al. "Bag of tricks for efficient text classification." FAIR 2016

Transfer Learning for Text

Similar to image: can we have word representations that are generic enough to **transfer** from one task to another?

Transfer Learning for Text

Similar to image: can we have word representations that are generic enough to **transfer** from one task to another?

Unsupervised / self-supervised learning of word representations

Transfer Learning for Text

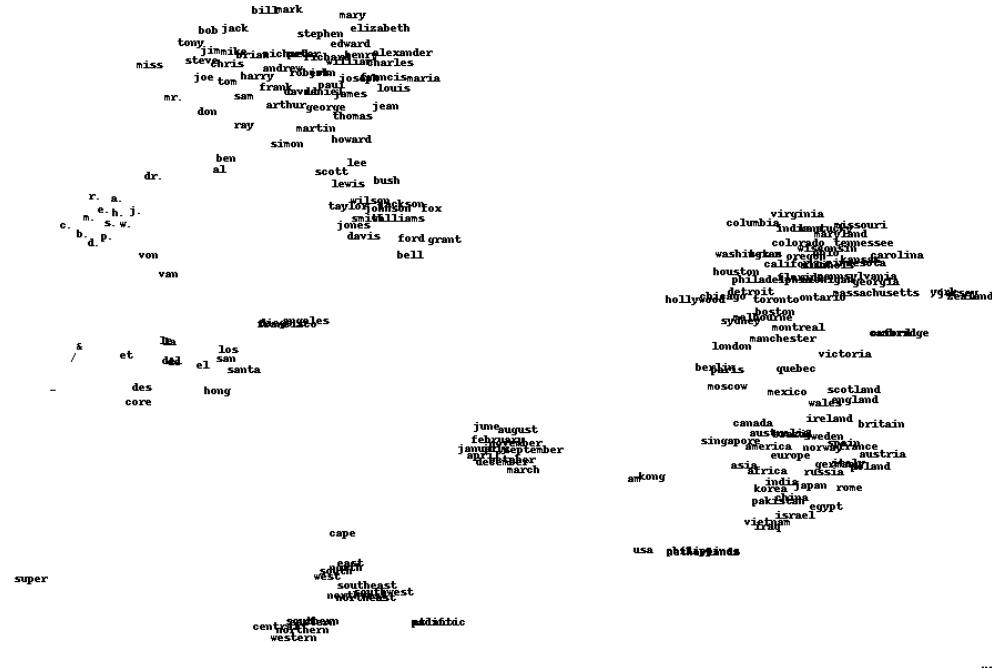
Similar to image: can we have word representations that are generic enough to **transfer** from one task to another?

Unsupervised / self-supervised learning of word representations

Unlabelled text data is almost infinite:

- Wikipedia dumps
- Project Gutenberg
- Social Networks
- Common Crawl

Word Vectors



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

Word2Vec

FRANCE	JESUS	XBOX	REDDISH	SCRATCHED	MEGABITS
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/S
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

Word2Vec

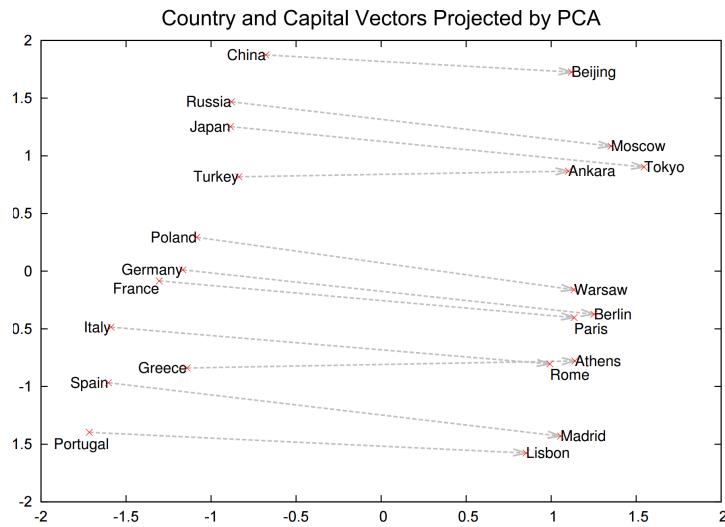
FRANCE	JESUS	XBOX	REDDISH	SCRATCHED	MEGABITS
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/S
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

Compositionality

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

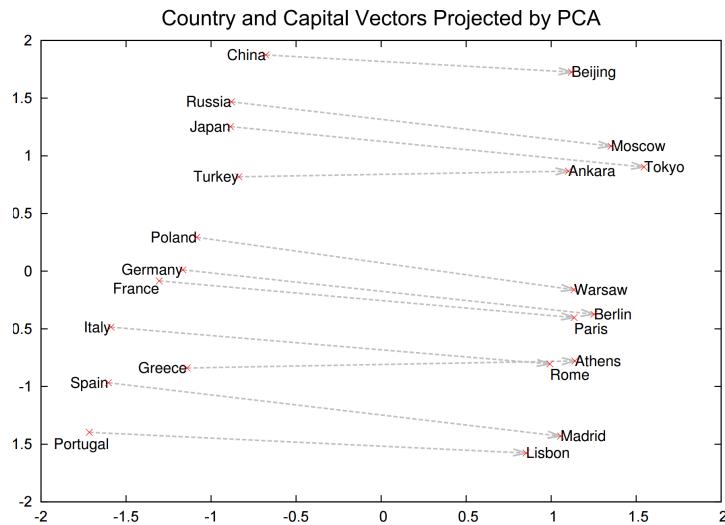
Colobert et al. 2011, Mikolov, et al. 2013

Word Analogies



Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." NIPS 2013

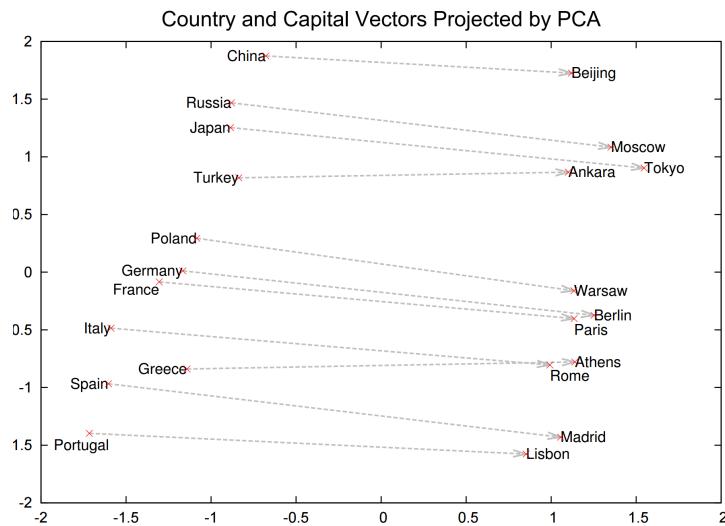
Word Analogies



- Linear relations in Word2Vec embeddings

Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." NIPS 2013

Word Analogies



- Linear relations in Word2Vec embeddings
- Many come from text structure (e.g. Wikipedia)

Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." NIPS 2013

Self-supervised training

Distributional Hypothesis (Harris, 1954): "*words are characterised by the company that they keep*"

Main idea: learning word embeddings by **predicting word contexts**

Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." NIPS 2013

Self-supervised training

Distributional Hypothesis (Harris, 1954): “*words are characterised by the company that they keep*”

Main idea: learning word embeddings by **predicting word contexts**

Given a word e.g. “carrot” and any other word $w \in V$ predict probability $P(w|\text{carrot})$ that w occurs in the context of “carrot”.

Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." NIPS 2013

Self-supervised training

Distributional Hypothesis (Harris, 1954): "*words are characterised by the company that they keep*"

Main idea: learning word embeddings by **predicting word contexts**

Given a word e.g. "carrot" and any other word $w \in V$ predict probability $P(w|\text{carrot})$ that w occurs in the context of "carrot".

- **Unsupervised / self-supervised:** no need for class labels.
- (Self-)supervision comes from **context**.
- Requires a lot of text data to cover rare words correctly.

Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." NIPS 2013

Word2Vec: CBoW

CBoW: representing the context as **Continuous Bag-of-Word**

Self-supervision from large unlabeled corpus of text: *slide over an anchor word and its context:*

the carrot is a root vegetable, usually orange

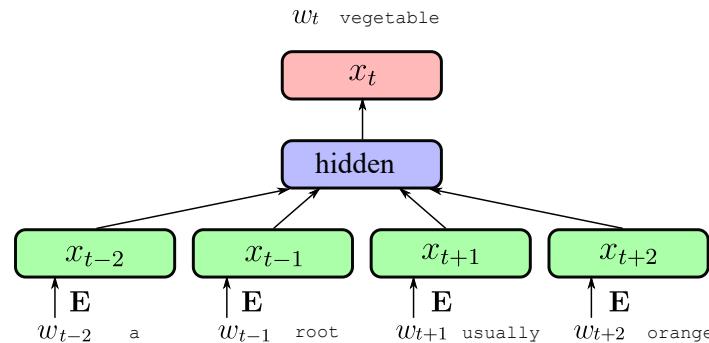
Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." NIPS 2013

Word2Vec: CBow

CBow: representing the context as **Continuous Bag-of-Word**

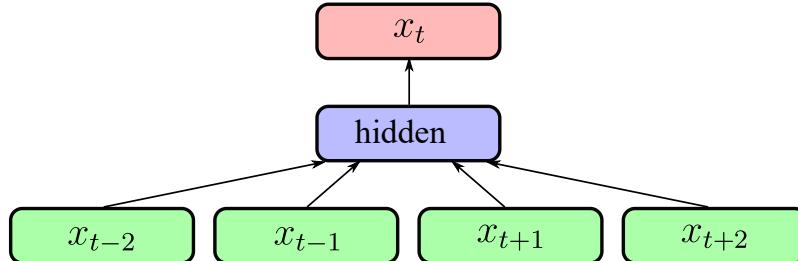
Self-supervision from large unlabeled corpus of text: *slide* over an anchor word and its context:

the carrot is a root vegetable, usually orange



Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." NIPS 2013

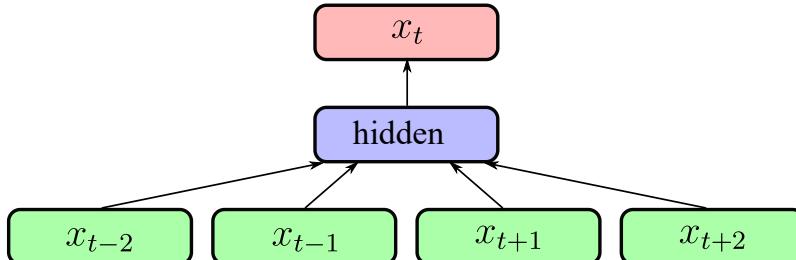
Word2Vec: Details



- Similar as supervised CBoW (e.g. fastText) with $|V|$ classes

Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." NIPS 2013

Word2Vec: Details

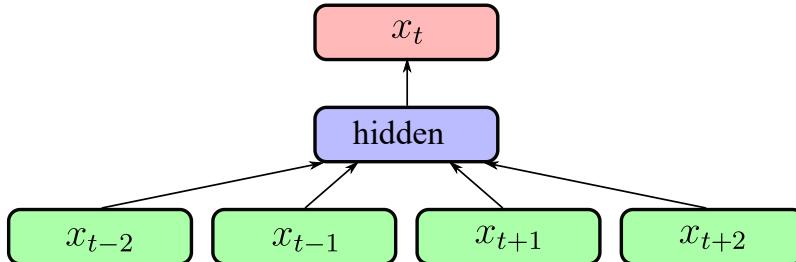


- Similar as supervised CBoW (e.g. fastText) with $|V|$ classes
- Use **Negative Sampling**: sample *negative* words at random instead of computing the full softmax. See:

<http://sebastianruder.com/word-embeddings-softmax/index.html>

Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." NIPS 2013

Word2Vec: Details

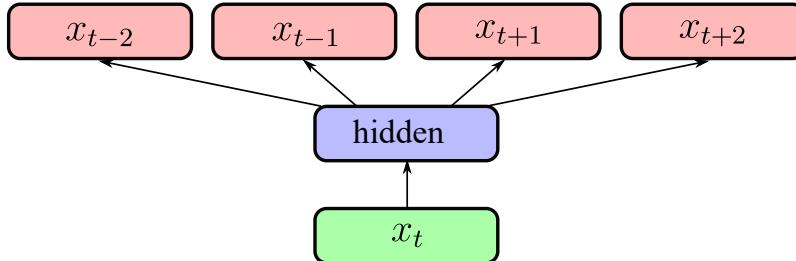


- Similar as supervised CBoW (e.g. fastText) with $|V|$ classes
- Use **Negative Sampling**: sample *negative* words at random instead of computing the full softmax. See:
<http://sebastianruder.com/word-embeddings-softmax/index.html>
- Large impact of **context size**

Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." NIPS 2013

Word2Vec: Skip Gram

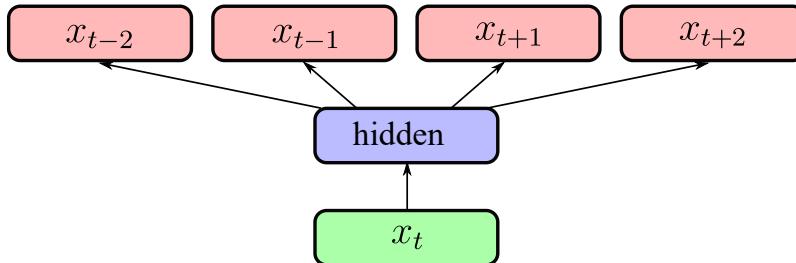
a



- Given the central word, predict occurrence of other words in its context.

Word2Vec: Skip Gram

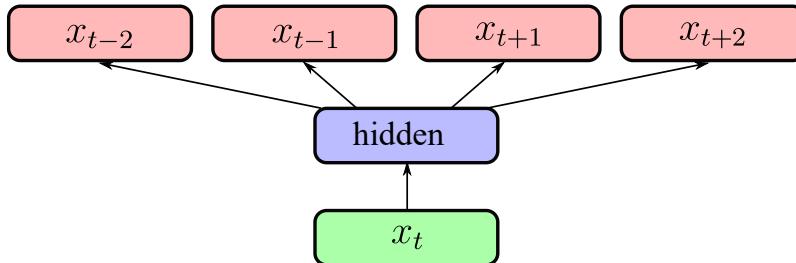
a



- Given the central word, predict occurrence of other words in its context.
- Widely used in practice

Word2Vec: Skip Gram

a



- Given the central word, predict occurrence of other words in its context.
- Widely used in practice
- Again **Negative Sampling** is used as a cheaper alternative to full softmax.

Evaluation and Related methods

Always difficult to evaluate unsupervised tasks

- WordSim (Finkelstein et al.)
- SimLex-999 (Hill et al.)
- Word Analogies (Mikolov et al.)

Evaluation and Related methods

Always difficult to evaluate unsupervised tasks

- WordSim (Finkelstein et al.)
- SimLex-999 (Hill et al.)
- Word Analogies (Mikolov et al.)

Other popular method: **GloVe** (Socher et al.)

<http://nlp.stanford.edu/projects/glove/>

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global Vectors for Word Representation." EMNLP. 2014

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 self-supervised 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 self-supervised embeddings** (transfer learning from Glove, word2vec or fastText embeddings)
- If **large training data** with labels, directly learn task-specific embedding with methods 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 self-supervised embeddings** (transfer learning from Glove, word2vec or fastText embeddings)
- If **large training data** with labels, directly learn task-specific embedding with methods 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 self-supervised embeddings** (transfer learning from Glove, word2vec or fastText embeddings)
- If **large training data** with labels, directly learn task-specific embedding with methods 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 self-supervised embeddings** (transfer learning from Glove, word2vec or fastText embeddings)
- If **large training data** with labels, directly learn task-specific embedding with methods 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.

Word Embeddings no long state of the art for NLP tasks: BERT-style pretraining of deep transformers with sub-word tokenization is now used everywhere.

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_θ 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_{θ} 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

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

p_{θ} 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

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

p_{θ} 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: $p(\text{Target}|\text{Source})$

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

Conditional Language Models

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

Translation: $p(\text{Target}|\text{Source})$

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

Model the output word by word:

$p_\theta(w_0|\text{Source})$

Conditional Language Models

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

Translation: $p(\text{Target}|\text{Source})$

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

Model the output word by word:

$$p_{\theta}(w_0|\text{Source}) \cdot p_{\theta}(w_1|w_0, \text{Source}) \cdot \dots$$

Conditional Language Models

Question Answering / Dialogue:

$$p(\text{Answer} | \text{Question}, \text{Context})$$

- *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:

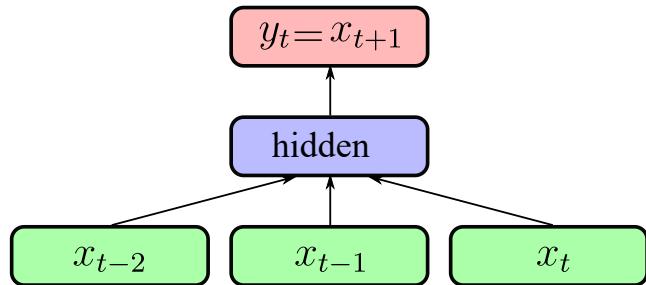
$$p(\text{Answer} | \text{Question}, \text{Context})$$

- *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."

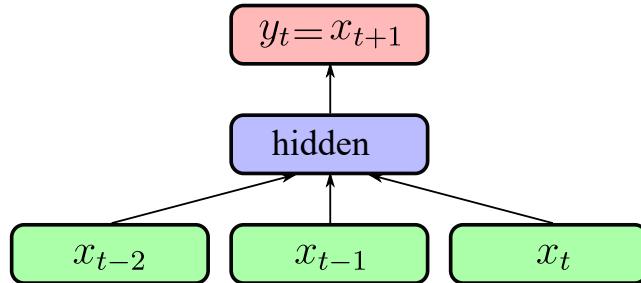
Image Captionning: $p(\text{Caption} | \text{Image})$

- Image is usually the 2048-d representation from a CNN

Simple Language Model



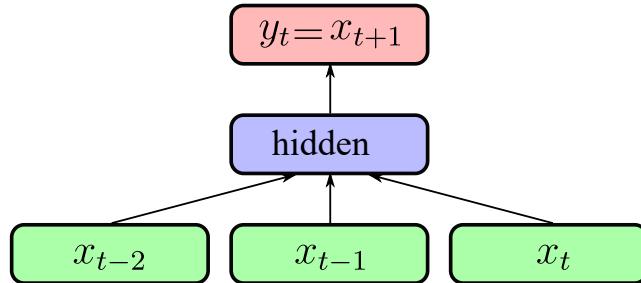
Simple Language Model



Fixed context size

- **Average embeddings:** (same as CBoW) no sequence information

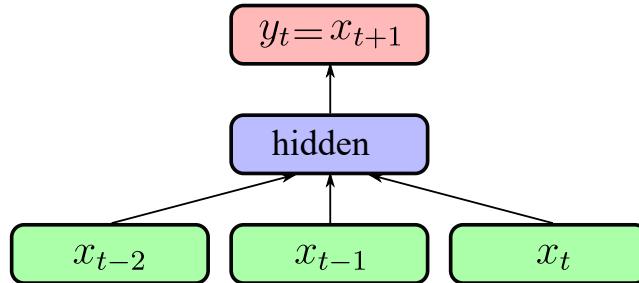
Simple Language Model



Fixed context size

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

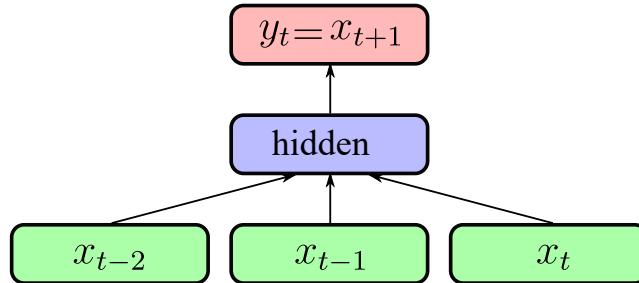
Simple Language Model



Fixed context size

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

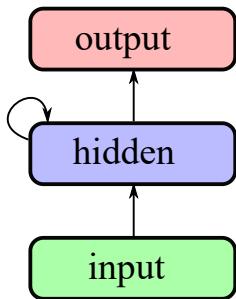
Simple Language Model



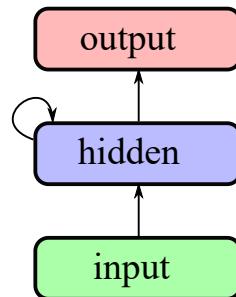
Fixed context size

- **Average embeddings:** (same as CBoW) no sequence information
- **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

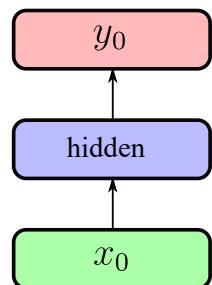
Recurrent Neural Network



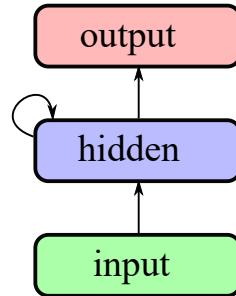
Recurrent Neural Network



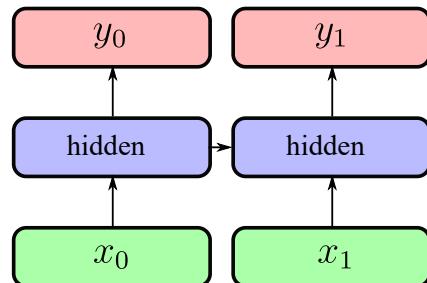
Unroll over a sequence (x_0, x_1, x_2) :



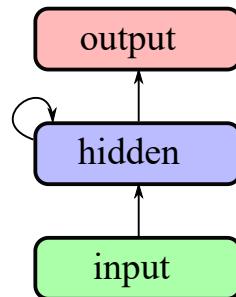
Recurrent Neural Network



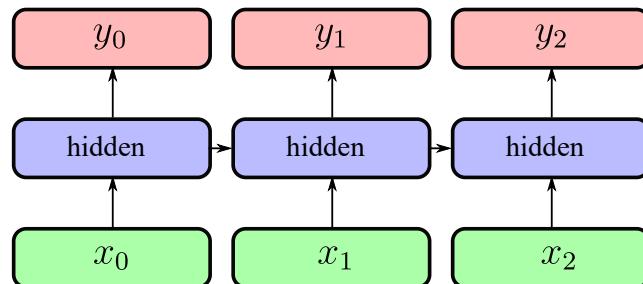
Unroll over a sequence (x_0, x_1, x_2) :



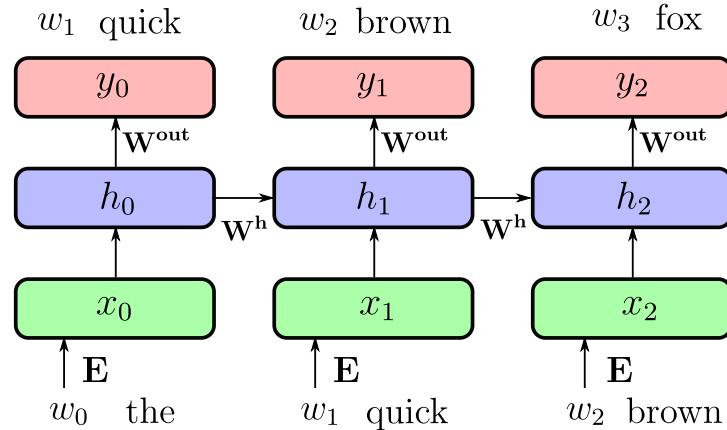
Recurrent Neural Network



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



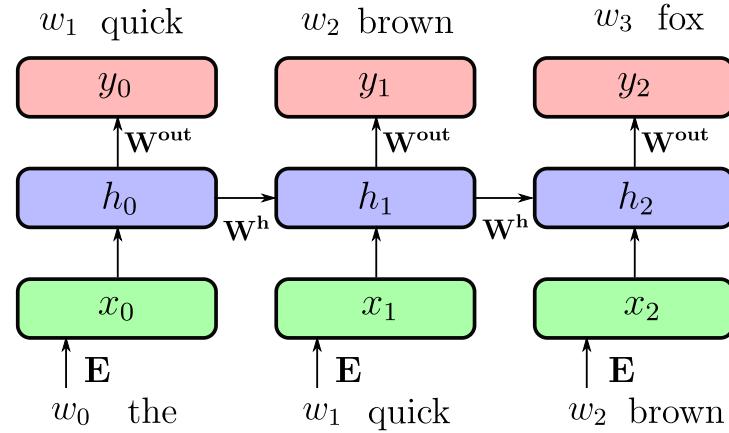
Language Modelling



input (w_0, w_1, \dots, w_t) sequence of words (1-hot encoded)

output $(w_1, w_2, \dots, w_{t+1})$ shifted sequence of words (1-hot encoded)

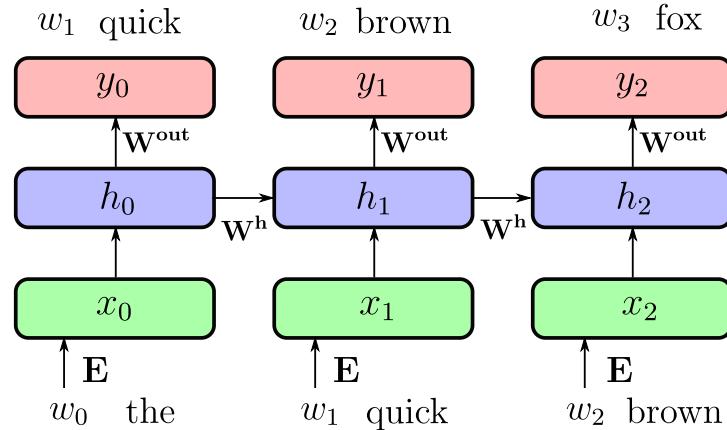
Language Modelling



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

input projection \mathbf{H}

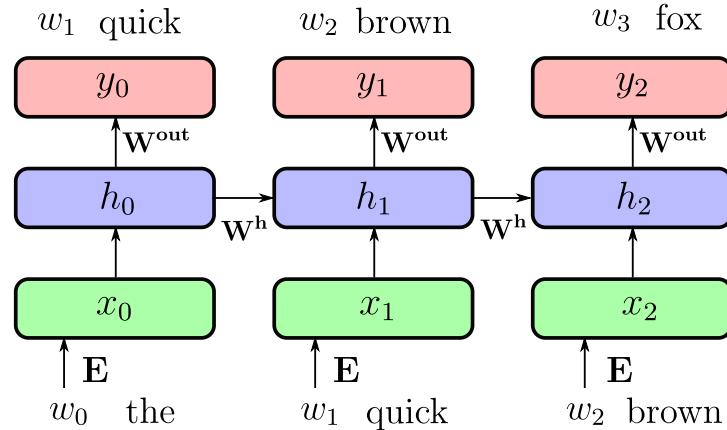
Language Modelling



$$x_t = \text{Emb}(w_t) = \mathbf{E}w_t \quad \text{input projection H}$$

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

Language Modelling

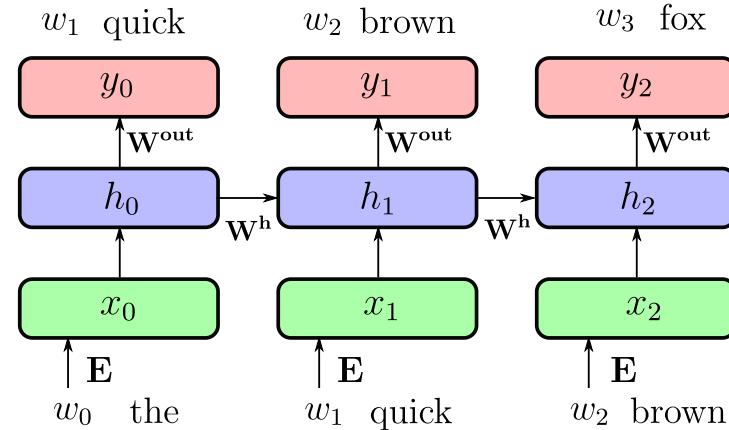


$$x_t = \text{Emb}(w_t) = \mathbf{E}w_t \quad \text{input projection H}$$

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

$$y = \text{softmax}(\mathbf{W}^o h_t + b^o) \quad \text{output projection K} = |\mathcal{V}|$$

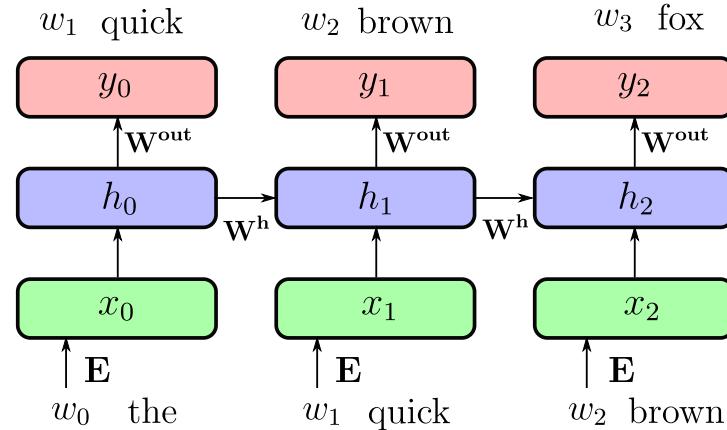
Recurrent Neural Network



Input embedding \mathbf{E}

$|V| \times H$

Recurrent Neural Network



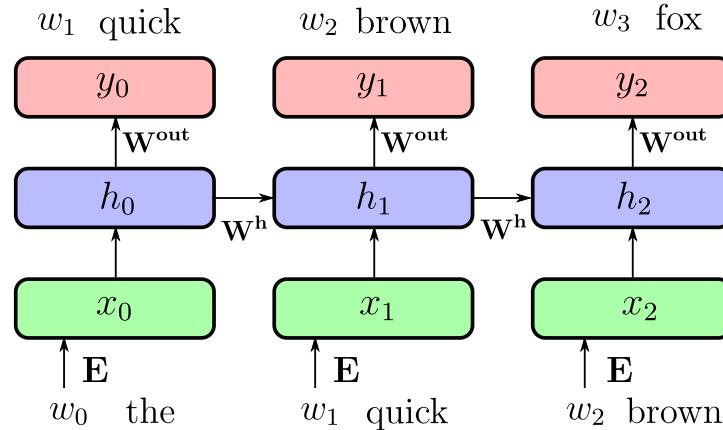
Input embedding \mathbf{E}

Recurrent weights \mathbf{W}^h

$|V| \times H$

$H \times H$

Recurrent Neural Network



Input embedding \mathbf{E}

$|V| \times H$

Recurrent weights \mathbf{W}^h

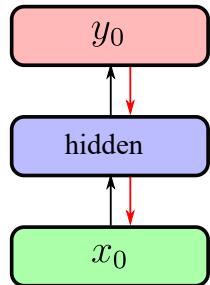
$H \times H$

Output weights \mathbf{W}^{out}

$H \times K = H \times |V|$

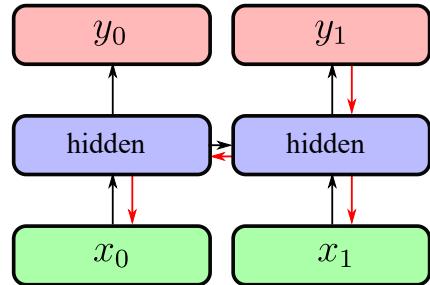
Backpropagation through time

Similar as standard backpropagation on unrolled network



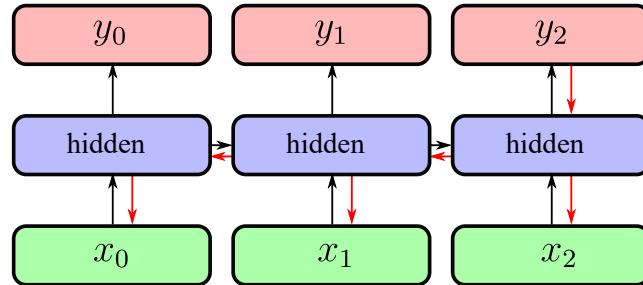
Backpropagation through time

Similar as standard backpropagation on unrolled network



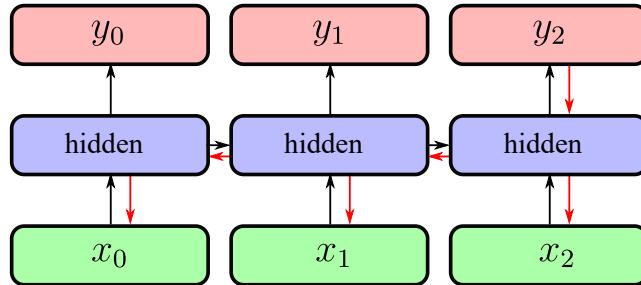
Backpropagation through time

Similar as standard backpropagation on unrolled network



Backpropagation through time

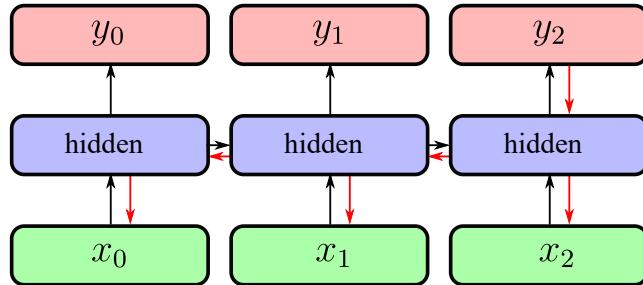
Similar as standard backpropagation on unrolled network



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

Backpropagation through time

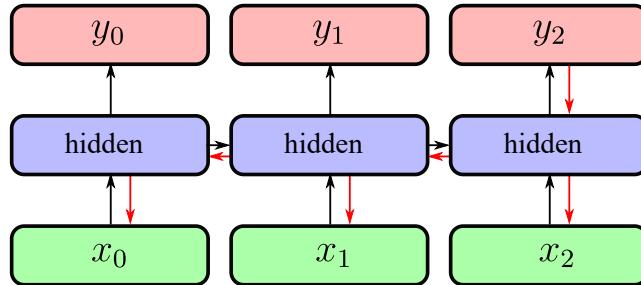
Similar as standard backpropagation on unrolled network



- 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

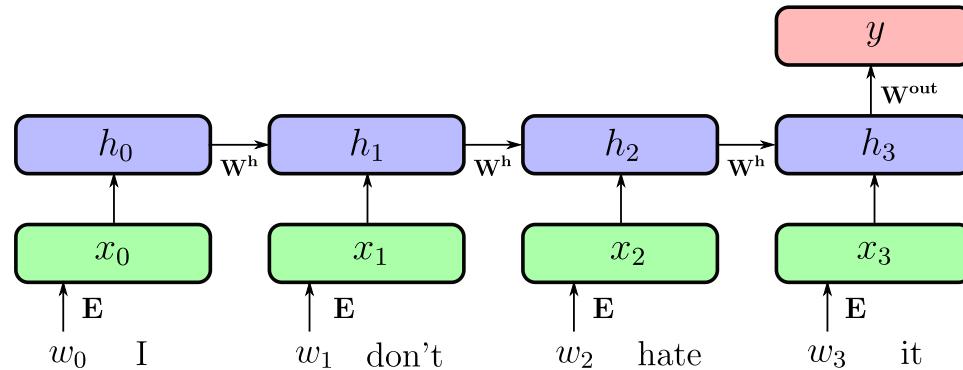
Backpropagation through time

Similar as standard backpropagation on unrolled network



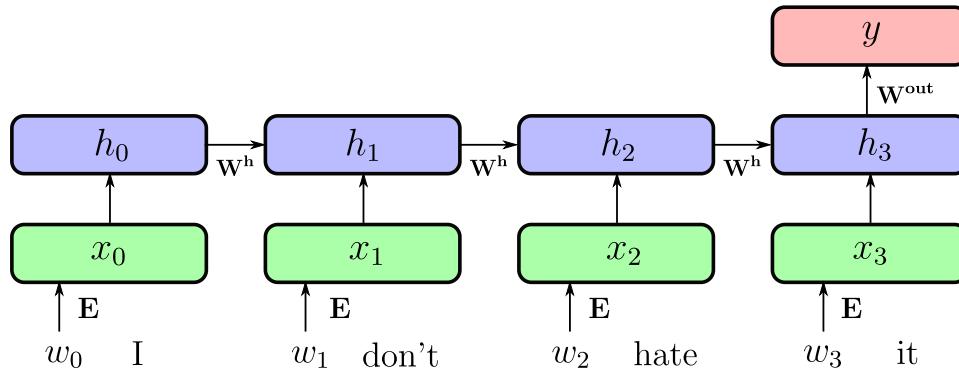
- Similar as training **very deep networks** with tied parameters
- Example between x_0 and y_2 : W^h is used twice
- Usually truncate the backprop after T timesteps
- Difficulties to train long-term dependencies

Other uses: Sentiment Analysis



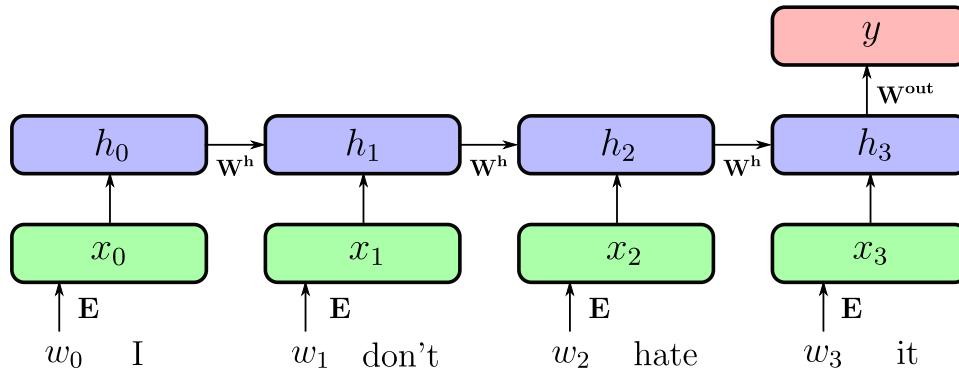
- Output is sentiment (1 for positive, 0 for negative)

Other uses: Sentiment Analysis



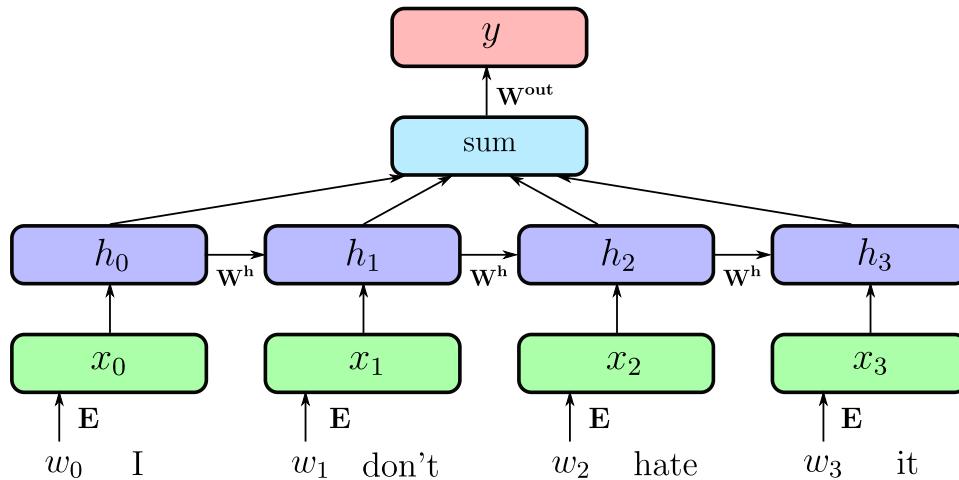
- Output is sentiment (1 for positive, 0 for negative)
- Very dependent on words order

Other uses: Sentiment Analysis



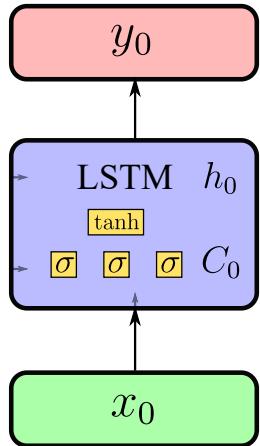
- Output is sentiment (1 for positive, 0 for negative)
- Very dependent on words order
- Very flexible network architectures

Other uses: Sentiment analysis



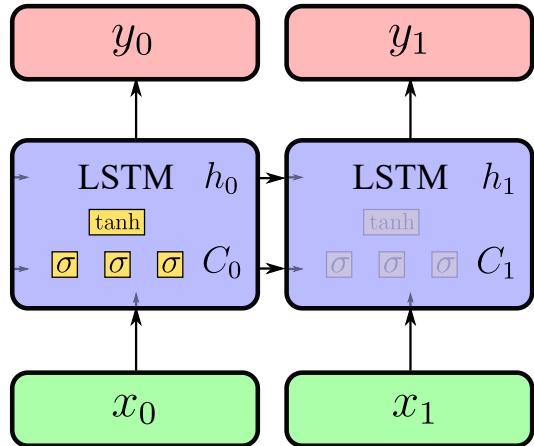
- Output is sentiment (1 for positive, 0 for negative)
- Very dependent on words order
- Very flexible network architectures

LSTM



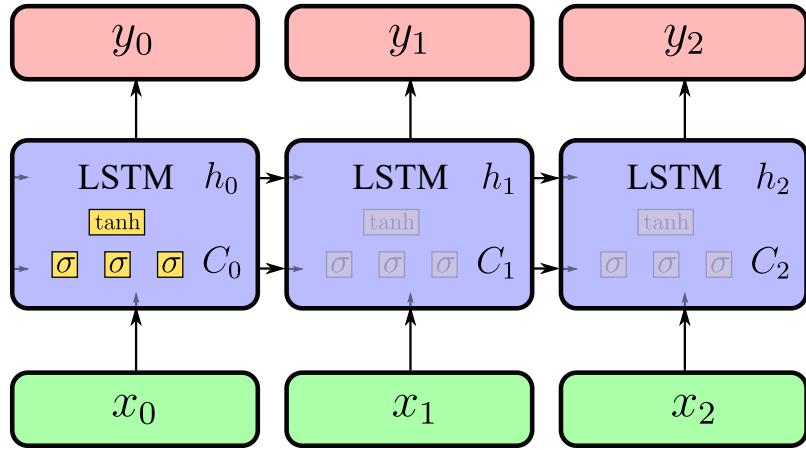
Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation
1997

LSTM



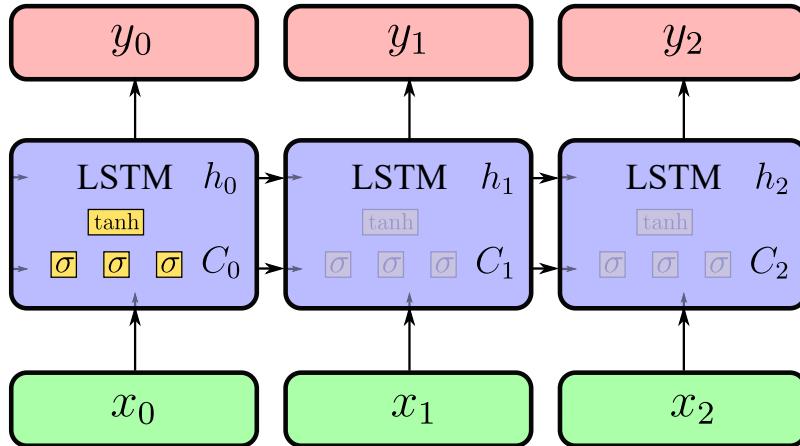
Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation
1997

LSTM



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

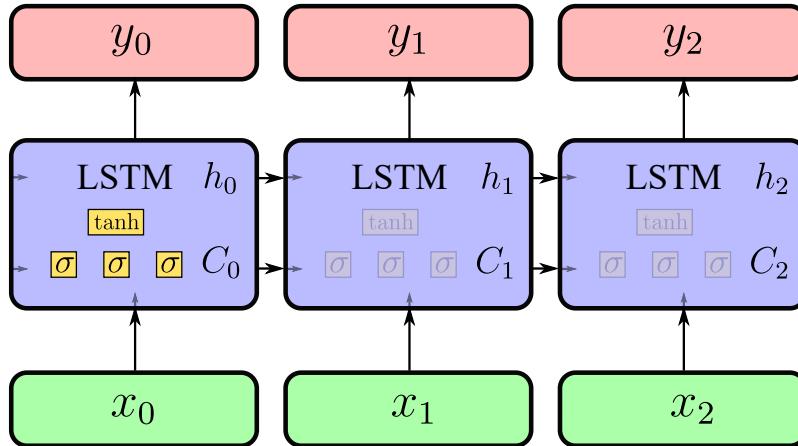
LSTM



- 4 times more parameters than RNN

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

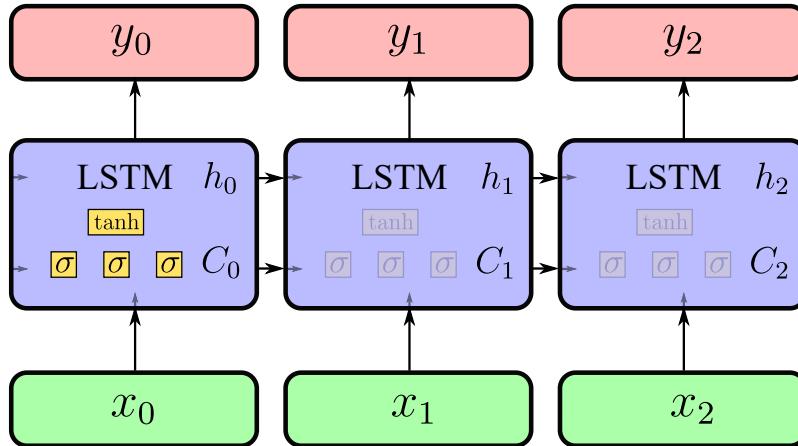
LSTM



- 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 SOTA in many sequence learning problems

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

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

Update gate \mathbf{H}

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

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

Update gate \mathbf{H}

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

Forget gate \mathbf{H}

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

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

Update gate \mathbf{H}

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

Forget gate \mathbf{H}

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

Cell candidate \mathbf{H}

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

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

Update gate H

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

Forget gate H

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

Cell candidate H

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

Cell output H

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

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

Update gate \mathbf{H}

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

Forget gate \mathbf{H}

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

Cell candidate \mathbf{H}

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

Cell output \mathbf{H}

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

Output gate \mathbf{H}

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

$\mathbf{u} = \sigma(\mathbf{W}^{\mathbf{u}} \cdot h_{t-1} + \mathbf{I}^{\mathbf{u}} \cdot x_t + b^u)$	Update gate \mathbf{H}
$\mathbf{f} = \sigma(\mathbf{W}^{\mathbf{f}} \cdot h_{t-1} + \mathbf{I}^{\mathbf{f}} \cdot x_t + b^f)$	Forget gate \mathbf{H}
$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}^{\mathbf{c}} \cdot h_{t-1} + \mathbf{I}^{\mathbf{c}} \cdot x_t + b^c)$	Cell candidate \mathbf{H}
$\mathbf{c}_t = \mathbf{f} \odot \mathbf{c}_{t-1} + \mathbf{u} \odot \tilde{\mathbf{c}}_t$	Cell output \mathbf{H}
$\mathbf{o} = \sigma(\mathbf{W}^{\mathbf{o}} \cdot h_{t-1} + \mathbf{I}^{\mathbf{o}} \cdot x_t + b^o)$	Output gate \mathbf{H}
$\mathbf{h}_t = \mathbf{o} \odot \tanh(\mathbf{c}_t)$	Hidden output \mathbf{H}

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

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

Update gate H

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

Forget gate H

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

Cell candidate H

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

Cell output H

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

Output gate H

$$\mathbf{h}_t = \mathbf{o} \odot \tanh(\mathbf{c}_t)$$

Hidden output H

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

Output K

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

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

Update gate \mathbf{H}

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

Forget gate \mathbf{H}

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

Cell candidate \mathbf{H}

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

Cell output \mathbf{H}

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

Output gate \mathbf{H}

$$\mathbf{h}_t = \mathbf{o} \odot \tanh(\mathbf{c}_t)$$

Hidden output \mathbf{H}

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

Output \mathbf{K}

$$W^u, W^f, W^c, W^o$$

Recurrent weights $\mathbf{H} \times \mathbf{H}$

$$I^u, I^f, I^c, I^o$$

Input weights $\mathbf{N} \times \mathbf{H}$

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

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

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 to 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 to 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.