Neural Networks for Language Processing

[DAT640] Information Retrieval and Text Mining

Petra Galuscakova
University of Stavanger

September 4, 2024

In this module

1. Language Models
2. Embeddings
3. Deep Neural Networks

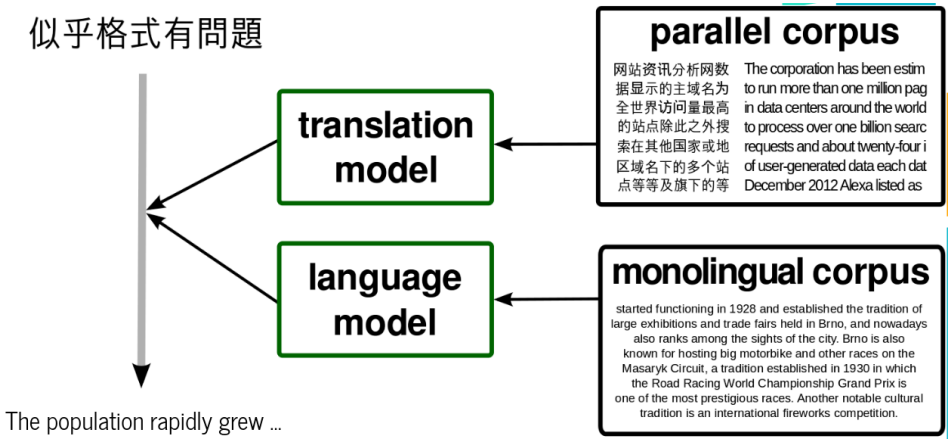
Language Models

Language Model

- A language model is a probability distribution over sequences of words [1].
- P(Twinkle twinkle little star, how I wonder what you are.) = 0.99
  P(Twinkle twinkle little moon, how I wonder what you are.) = 0.75
  P(Twinkle twinkle little star, how I what you are.) = 0.3
  P(Are you what I wonder I how star, little twinkle, twinkle.) = 0.02

[1] Jurafsky and Martin: Speech and Language Processing, 2023

Language Models in Machine Translation[2]

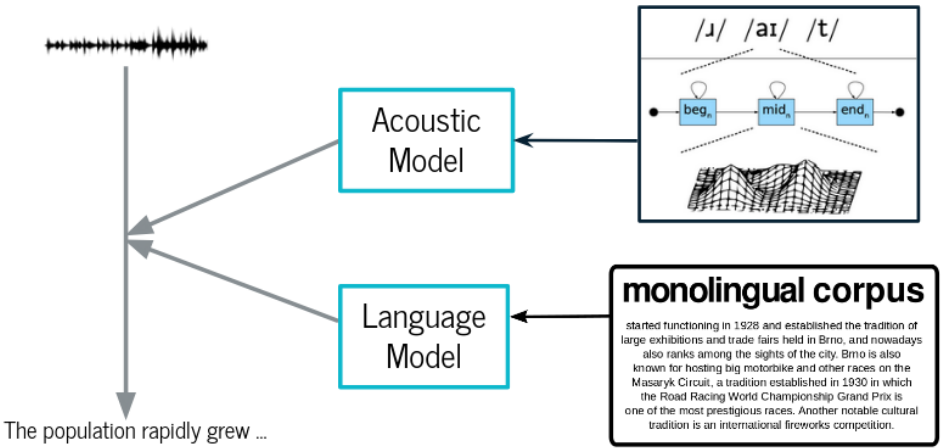


[2] nlp.fi.muni.cz

Language Models in Machine Translation

- Requires several models:
- Translation model contains probabilities of the translations of words. These might be learned from a parallel corpus. Words in the sentences in both languages are aligned and the numbers of translations are counted.
  Haus — house (p=0.8), building (p=0.16), home (p=0.02), household (p=0.015), shell (p=0.005).
- Reordering model
- Language model knows the probability of an occurrence of each word in the target language.

Language Models in Speech Recognition



Language Models in Speech Recognition

- Acoustic Model is a statistical model, that maps the features extracted from the signal to phonetic or sub-word representations to capture the acoustic-textual relationship. It contains the probabilities that the phonemes were generated from the recognized acoustic features.
- Pronunciation model, which is usually created by human experts. It maps phonemes with text.
- Language model knows the probability of an occurrence of each word in the target language.

N-grams

- Which word will follow?
  Please turn in your homework
- N-gram is a sequence of n words
- n=1 -> unigram [please, turn, in, your, homework]
- n=2 -> bigram [please turn, turn in, in your, your homework]
- n=3 -> trigram [please turn in, turn in your, in your homework]

N-gram Probabilities

- Which word will follow?
  Please turn in your homework ...
- Calculate P(w|h): the probability of a word w given some history h.
- Estimate this probability from relative frequency counts
- $P(\text{by} \mid \text{please turn in your homework}) = \frac{C(\text{please turn in your homework by})}{C(\text{please turn in your homework})}$
- C(Please turn in your homework on) = 12
  C(Please turn in your homework by) = 8
  -> C(Please turn in your homework) = 20
- -> P(by | please turn your homework) = 8 / 20 = 0.4
  -> P(on | please turn your homework) = 12 / 20 = 0.6
  Which word will follow?
  Please turn in your homework on

Smoothing

- When dealing with n-gram models, smoothing refers to the practice of adjusting empirical probability estimates to account for insufficient data, i.e. unseen n-grams won't have a zero probability as this might cause problems in applied multiplications.
- E.g. Laplace smoothing is the assumption that each n-gram in a corpus occurs exactly one more time than it actually does.

Exercise: N-grams

https://books.google.com/ngrams/

Embeddings

Language Models vs. Embeddings

- Language Models are a comprehensive NLP structures that 'understand' and generate human-like text. They help to predict the next word in a sentence on basis of the context extracted from the previous words.
- Embeddings are a method to transform words/phrases into numerical vectors.

Discrete (Sparse) Text Representation

- Sparse representation has the length of the vocabulary and each element of the vector corresponds to a term in the vocabulary
- The *i*-th component of the vector encodes a discrete value, e.g., the presence/absence of a word or its number of occurrences in the represented text (query or document)
- One-hot-vector: the vector consists of 0s in all cells with the exception of a single 1 in a cell used uniquely to identify the word.
- Sparse representations are based on the bag of words (BOW) model and the position of the word in the text is ignored in the representation

ID	term
1	discrete
2	sparse
3	document
4	text
5	word
6	this
	...

Vocabulary

$d =$ "this is sparse text"

$d = [t_6, OOV, t_2, t_4]$

$\vec{d} = \langle 0, 1, 0, 1, 0, 1, \dots \rangle$

Distributional semantics

You shall know a word by the company it keeps.

(J. R. Firth, A synopsis of linguistic theory (1957))

Distributional semantics

- Words that occur in the same context tend to have a similar meaning, e.g. book, volume, novel
- The meaning of words can be inferred by their usage together with other words in existing texts
- The *context* of a word is the set of words that appear within a fixed-size window. They are called *context-words*.

Distributional representation

- A word (token) can be represented as a vector of d dimensions
- d is much smaller that the size of the vocabulary
- This d-dimensional vector is called *distributional representation* or *word embedding*

Word-context matrix[3]

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

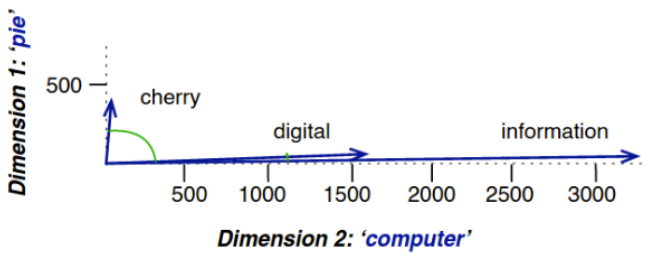
Word-context matrix is a co-occurence matrix with W rows (words) and C columns (contexts), where f_{ij} gives the number of times word w_i occurs in context c_j
We distinguish different types of similarity:

- first-order co-occurence - words appears nearby each other, e.g., *write* is a first-order associate of *book* or *poem*
- second-order co-occurence - words have similar neighbors, e.g., *write* is a second-order associate of words like *say* or *remark*

[3] `https://web.stanford.edu/~jurafsky/slp3/`

From Word-context to Embeddings

- Sencond-order co-occurence can be used to represent words
- Each row forms an embedding representation of the word: digital = (0 …., 1670, 1683, 85, 5, 4, ….)



Characteristics of word embeddings

- Components of word embeddings are rarely 0: they are real numbers, and can also have negative values
- Components of distributional representation are not mutually exclusive. More than one component can be "active" at the same time.

$\vec{w} =$

0.15	0.07	0.83	0.46	...	0.02

- Word embeddings are often referred to as *dense representations*

Word embeddings - visualization

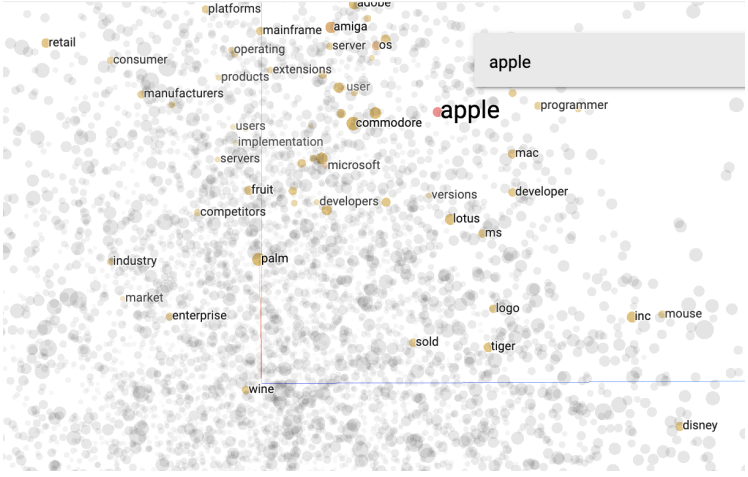
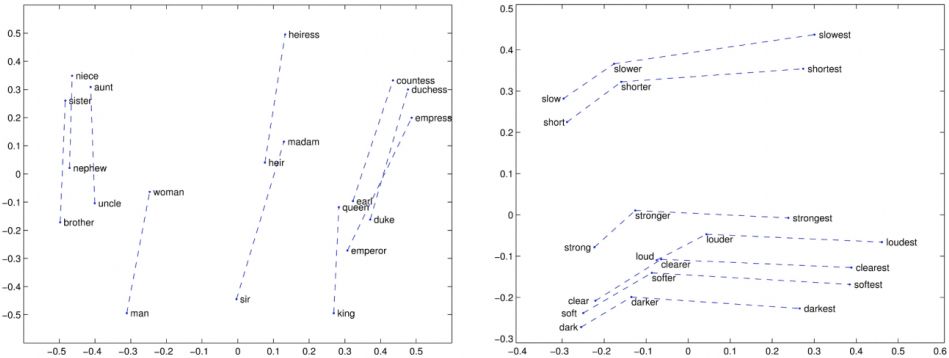


Figure: Tensorflow Embedding Projector [4]

[4] https://projector.tensorflow.org/

Analogy problem[5]



Analogy/relational similarity: Embeddings are able to capture relational meanings. Analogy problem can be generalized to the form *X is to Y as X* is to what?*.

[5] `https://web.stanford.edu/~jurafsky/slp3/`

Bias in word embeddings

- Embeddings reproduce the implicit biases and stereotypes that are latent in the text
- Bias visible in relational similarity:
  - *man:woman::king:queen*
  - *father:mother::doctor:nurse*
- Allocational harm - situation when system allocates resources unfairly to different groups
- Representational harm - situation when system demean or ignore some social groups
- Embeddings don't only reflect the statistics of their input, but they also amplify bias $\rightarrow$biased terms become more biased in embedding space than they were in the input text statistics
- Debiasing is still an open problem

Deep Neural Networks

Neural Networks

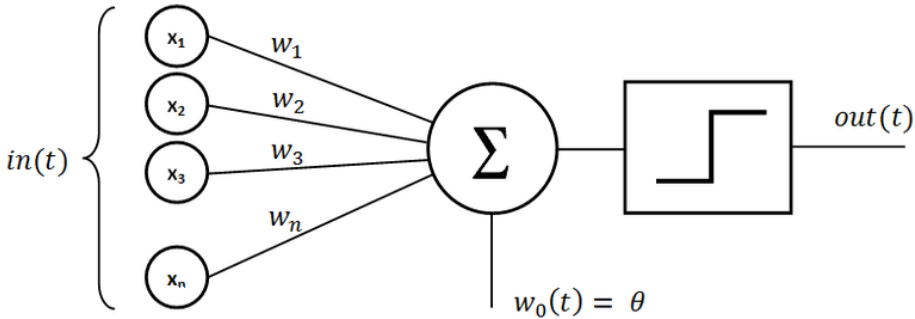
- Neural Network is a network of artificial neurons
- A neuron has some (weighted) inputs, when the input is high enough, it fires a signal
- Deep Neural Network: neural networks with many layers, typically uses back-propagation for training

Perceptron

- The perceptron takes multiple input features, each input feature represents a characteristic or attribute of the input data (x_i)[6]
- Each input feature is associated with a weight (w_i), determining the significance of each input feature in influencing the perceptron's output. During training, these weights are adjusted to learn the optimal values.
- Summation Function: The perceptron calculates the weighted sum of its inputs using the summation function.
- Activation Function: The weighted sum is then passed through an activation function. Perceptron can uses Heaviside function which takes the summed values as input, compare them with the threshold and provides the output as 0 or 1.
- The bias ($w_0(t)$) allows the model to make adjustments that are independent of the input. It is an additional parameter that is learned during training.

[6] https://www.geeksforgeeks.org/what-is-perceptron-the-simplest-artificial-neural-network/

Single Neuron (Perceptron)



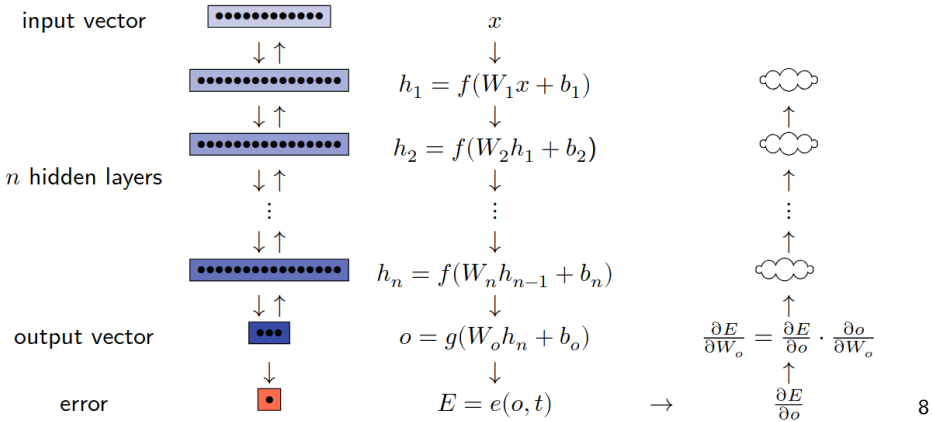
7

[7] https://commons.wikimedia.org

From a Perceptron to NN

- The perceptron's output is a weighted sum of the inputs that have been run through an activation function to decide whether or not the perceptron will fire:
  $z = w_1x_1 + w_2x_2 + ... + w_nx_n = X^TW$
  $out(t) = h(z)$ where h is the activation function
- The neurons are organized at the layers. The input neuron is connected to the hidden layer neuron, and the hidden layer neuron is connected to the output neuron, but the neurons of the same layer are not connected. When all the neurons in a layer are connected to every neuron of the previous layer, it is known as a fully connected layer or a dense layer.
- The output of the fully connected layer can be written as: $f_{W,b} = h(XW + b)$ where X is the input W is the weight for each inputs neurons and b is the bias and h is the activation function.

NN Structure

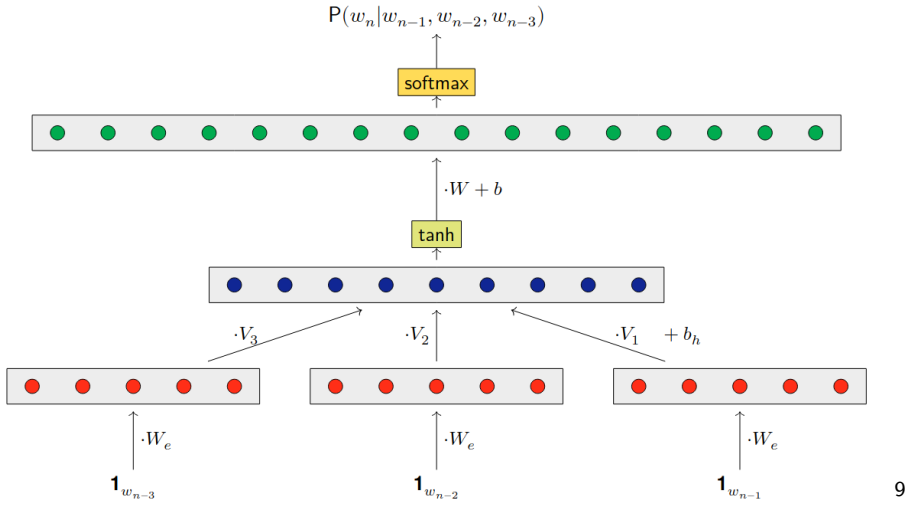


[8]Source: Jindrich Libovicky

NN Training

- During training, the perceptron's weights are adjusted to minimize the difference between the predicted output and the actual output.
- Parameter training is usually done using backpropagation.
- The first step of the backpropagation is the forward pass. Once the network's output is obtained, the error is calculated using a loss function. Common loss functions include Mean Squared Error (MSE) for regression tasks and Cross-Entropy Loss for classification tasks. The goal of training is to minimize this error.
- In the backward pass, the algorithm computes the gradient of the loss function with respect to each weight in the network to know how much each weight contributed to the overall error. This is done by applying the chain rule.
- The error is propagated backward through the network, from the output layer to the input layer. At each layer, the algorithm updates the weights by subtracting a fraction of the gradient from the current weights. This fraction is determined by the learning rate, a hyperparameter that controls the size of the weight updates.

Simple Neural LM



9

[9] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model.

Simple Neural LM

- One-hot vectors (**1**_$W = (0, ..., 0, 1, 0, ..., 0)$ are used on the input
- V is a table of learned word vector representations, so called word embeddings
- $h_1 = V_{w_{i-n}} \oplus V_{w_{i-n+1}} \oplus ... \oplus V_{w_{i-1}}$
- The task is a prediction of the following word
- Last layer is a probability distribution over the vocabulary
- Training objective: cross-entropy between the true (i.e., one-hot) distribution and estimated distribution

word2vec: static word embeddings

- Static word embeddings are global representations of the words, i.e., there is a single fixed word embedding for each term in the vocabulary. Static word embeddings map terms with multiple senses into an average or most common sense representation based on the training data used to compute the vectors.
- Word embeddings are a by-product of neural language models
- We can use a neural language model to create a word embedding, by using the model to predict the probability of appearance of a context-word in a window around the given word

word2vec: skip-gram

- Intuition:
  - Treat the target word and a neighboring context word as positive examples
  - Randomly sample other words in the lexicon to get negative samples
  - Use logistic regression to train a classifier to distinguish those two cases
  - Use the learned weights as the embeddings
- Skip-gram trains a logistic regression classifier to compute the probability that two words are 'likely to occur nearby in text'. This probability is computed from the dot product between the embeddings for the two words.

word2vec: skip-gram

- Maximize the probability of true context words w_{t-m}, w_{t-m+1}, ..., w_{t-1}, w_{t+1}, ..., w_{t+m-1}, w_{t+m} for each target word w_t:

$$J'(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P(w_{t+j}|w_t;\theta)$$

- Negative Log Likelihood:

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log P(w_{t+j}|w_t;\theta)$$

word2vec: negative sampling

The skip-gram model tries to shift embeddings so that the target embeddings are closer to (have a higher dot product with) context embeddings for nearby words and further from (lower dot product with) context embeddings for noise words that don't occur nearby

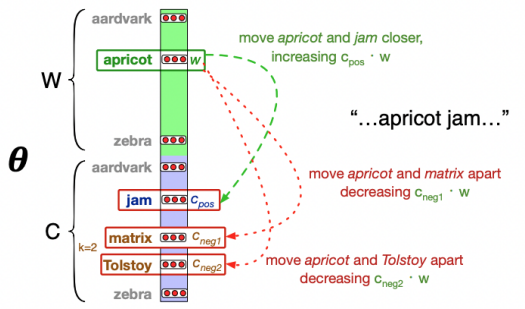


Figure: The skip-gram model [a]

[a] https://web.stanford.edu/~jurafsky/slp3/

word2vec: architecture

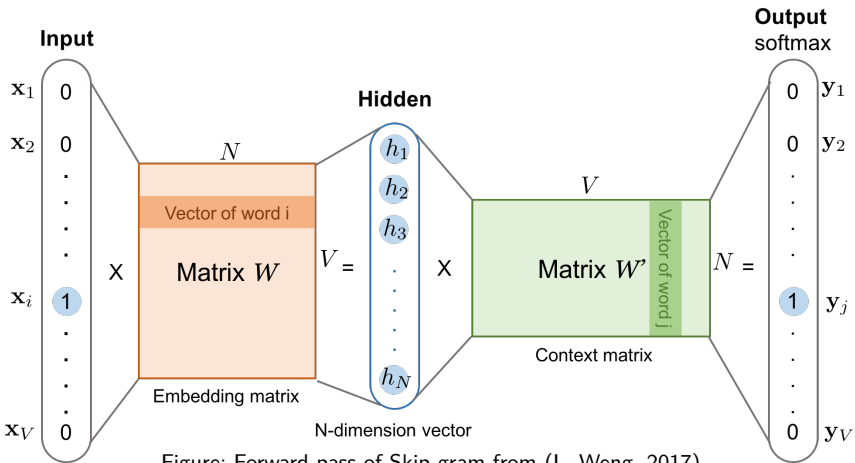


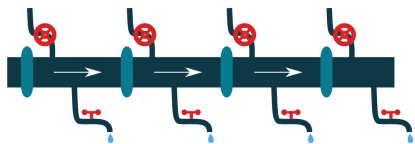
Figure: Forward pass of Skip-gram from (L. Weng, 2017).

Exercise

E6-1 Exploring Word2Vec with Gensim

RNN

- Recurrent Neural Networks (RNN): In every step some information goes in and some information goes out.
- They correspond well with the linear nature of the language. They are trained to work on sequential input and provide a sequential output (e.g. language modeling, machine translation, speech recognition, time series analysis)



[10]

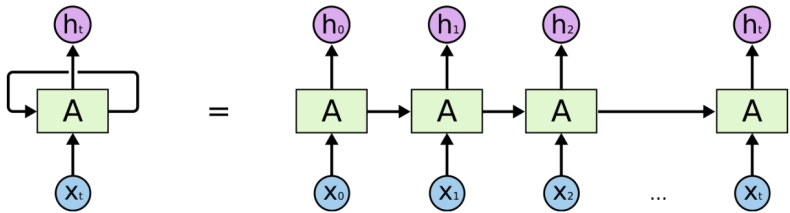
[10] Source: Jindrich Libovicky

RNN

- The recurrent connection enables RNNs to maintain internal memory, where the output of each step is fed back as an input to the next step, allowing the network to capture the information from previous steps and utilize it in the current step.
- The RNN takes an input vector X and the network generates an output vector y by scanning the data sequentially from left to right, with each time step updating the hidden state (i.e. internal memory)and producing an output.
- It shares the same parameters across all time steps.
- Backpropagation Through Time is a special variant of backpropagation used to train RNNs, where the error is propagated backward through time until the initial time step t=1.
- During backpropagation, gradients can become too small, leading to the vanishing gradient problem, or too large, resulting in the exploding gradient problem as they propagate backward through time.
- To address these problems, variations of RNN like Long-short term memory (LSTM) and Gated Recurrent Unit (GRU) networks have been introduced.

RNN

- Inputs: $x_1 ... x_T$ (word embeddings)
- Recurrent computation: $h_t = A(h_{t-1}, x_t)$

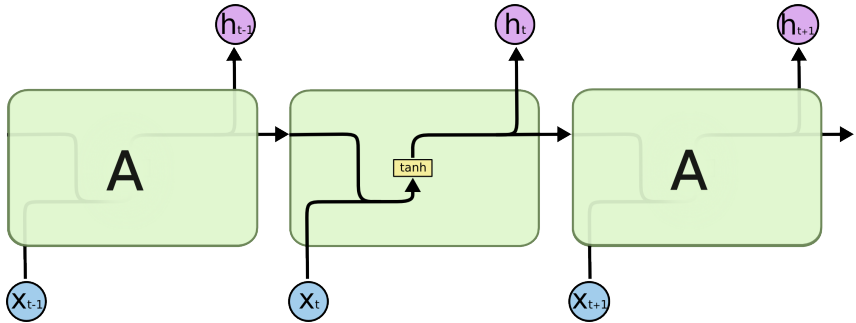


11

[11] Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs

Simple RNN

- Repeating module will have a very simple structure, such as a single tanh layer.



12

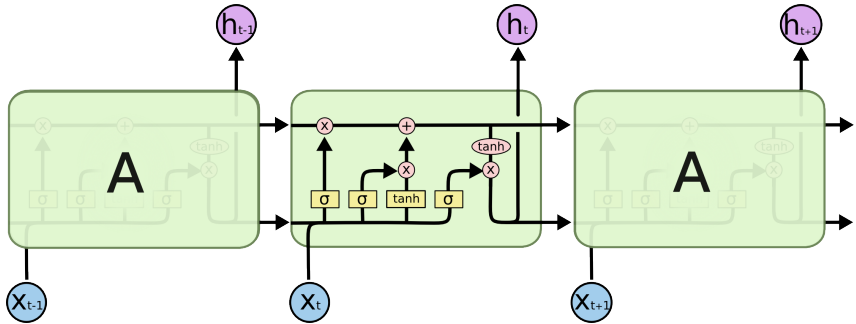
[12] Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs

Long Short-Term Memory Networks (LSTM)

- LSTMs are a special kind of RNN, capable of learning long-term dependencies
- LSTMs also have chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a specific way.
- The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

LSTM

- Yellow boxes are the neural networks.



13

[13] Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs

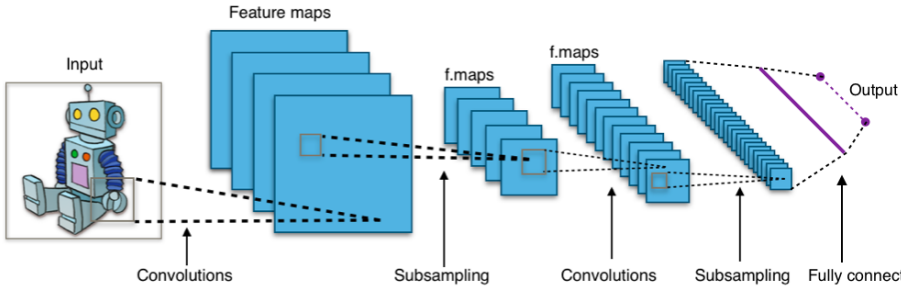
Long Short-Term Memory Networks (LSTM)

- LSTM does have the ability to remove or add information to the cell state, regulated by the gates.
- Gates are composed out of a sigmoid neural net layer and a pointwise multiplication operation.
- The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through.

CNN

- CNN are deep NN well suited for image processing
- The input layer represents the input image. It has three channels, corresponding to the red, green, and blue.
- Convolutional Layers contain the learned kernels (weights) which extract image features. The convolutional neuron performs an elementwise dot product with a unique kernel and the output of the previous layer's corresponding neuron.
- Pooling Layers (subsampling) decrease the spatial extent of the network. Can be done using Max-Pooling.

CNN



14

[14] https://en.wikipedia.org/wiki/Convolutional_neural_network

Demo: CNN Explainer

https://poloclub.github.io/cnn-explainer/

Summary

- N-gram Language Models
- Word embeddings
- Neural networks
- RNN, LSTM, CNN

Reading

- *Speech and Language Processing, Chapter 3: N-gram Language Models*, Dan Jurafsky and James H. Martin [15]
- *Speech and Language Processing, Chapter 3: Neural Networks*, Dan Jurafsky and James H. Martin [16]
- *Understanding LSTM Networks*, Christopher Olah[17]
- *CNN Explainer*, Wang, Zijie J. and Turko, Robert and Shaikh, Omar and Park, Haekyu and Das, Nilaksh and Hohman, Fred and Kahng, Minsuk and Chau, Duen Horng[18]

[15] `https://web.stanford.edu/~jurafsky/slp3/3.pdf`
[16] `https://web.stanford.edu/~jurafsky/slp3/7.pdf`
[17] `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`
[18] `https://poloclub.github.io/cnn-explainer/`