Natural Language Processing

December 9, 2016

Agenda

1. Model representation
2. Cost function
3. Gradient descent

Useful resources

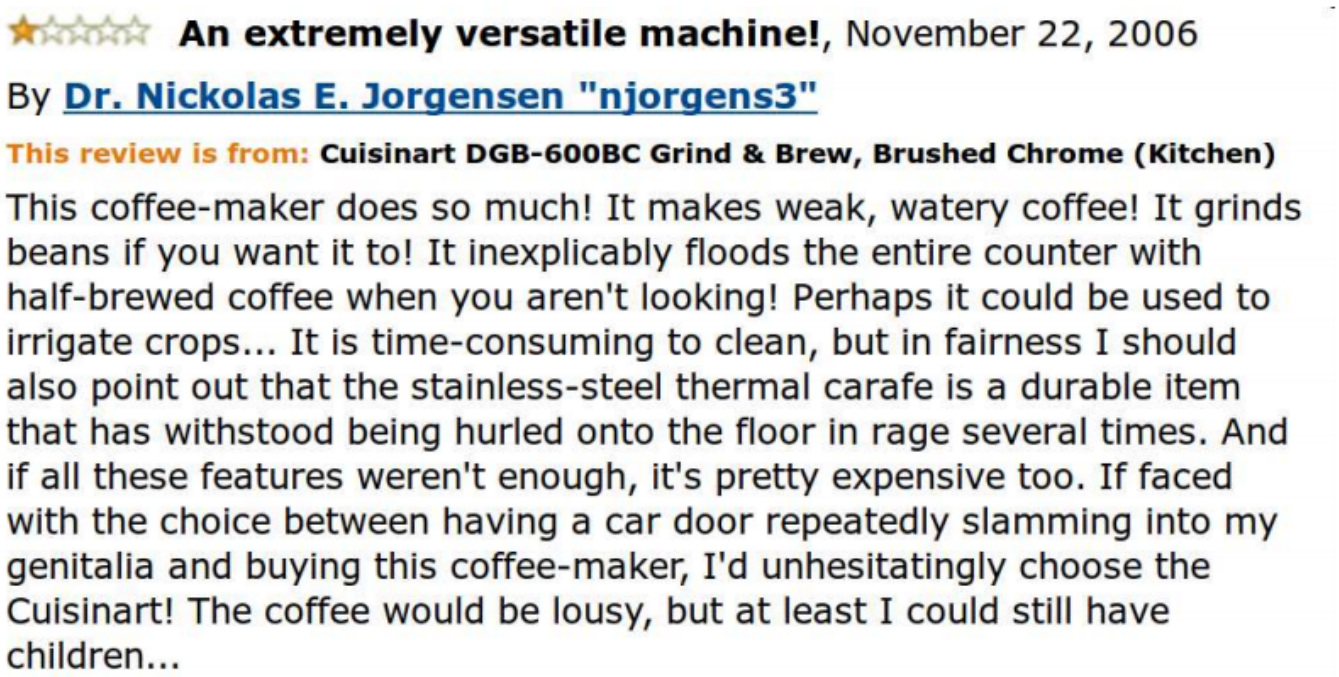
1. Stanford Deep learning for Natural Language Processing Course
2. Word2Vec original paper
3. Understanding LSTMs
4. NLP representation
5. NPL course from University of Michigan. (Coursera)

NLP tasks

1. Automatic summarization
2. Machine translation
3. Named entity recognition (NER)
4. Natural language generation
5. Part-of-speech tagging
6. Question answering
7. Sentiment analysis
8. Topic segmentation and recognition
9. Information retrieval (IR)

Sentiment classification

E.g. We need to find features of the text that could help predict number of stars.



★☆☆☆☆ **An extremely versatile machine!**, November 22, 2006

By **Dr. Nickolas E. Jorgensen "njorgens3"**

This review is from: Cuisinart DGB-600BC Grind & Brew, Brushed Chrome (Kitchen)

This coffee-maker does so much! It makes weak, watery coffee! It grinds beans if you want it to! It inexplicably floods the entire counter with half-brewed coffee when you aren't looking! Perhaps it could be used to irrigate crops... It is time-consuming to clean, but in fairness I should also point out that the stainless-steel thermal carafe is a durable item that has withstood being hurled onto the floor in rage several times. And if all these features weren't enough, it's pretty expensive too. If faced with the choice between having a car door repeatedly slamming into my genitalia and buying this coffee-maker, I'd unhesitatingly choose the Cuisinart! The coffee would be lousy, but at least I could still have children...

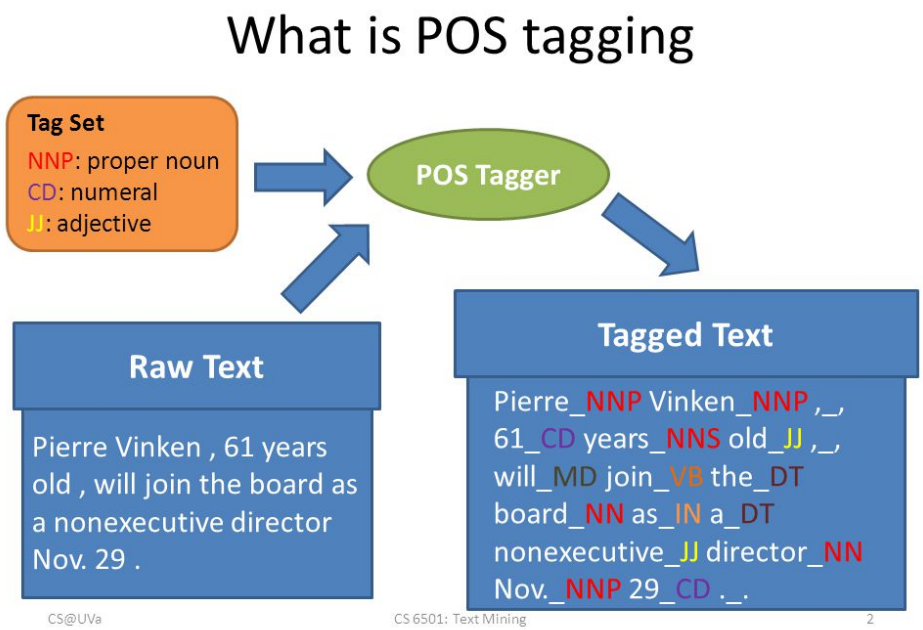
Other text categorization tasks

1. Spam detection
2. Topic modeling
3. Finding interesting to this user topics
4. Sentiment analyses

Part of Speech Tagging

We could treat tagging as a token classification problem

1. Tag each word independently given features of context
2. And features of the word's spelling (suffixes, capitalization)



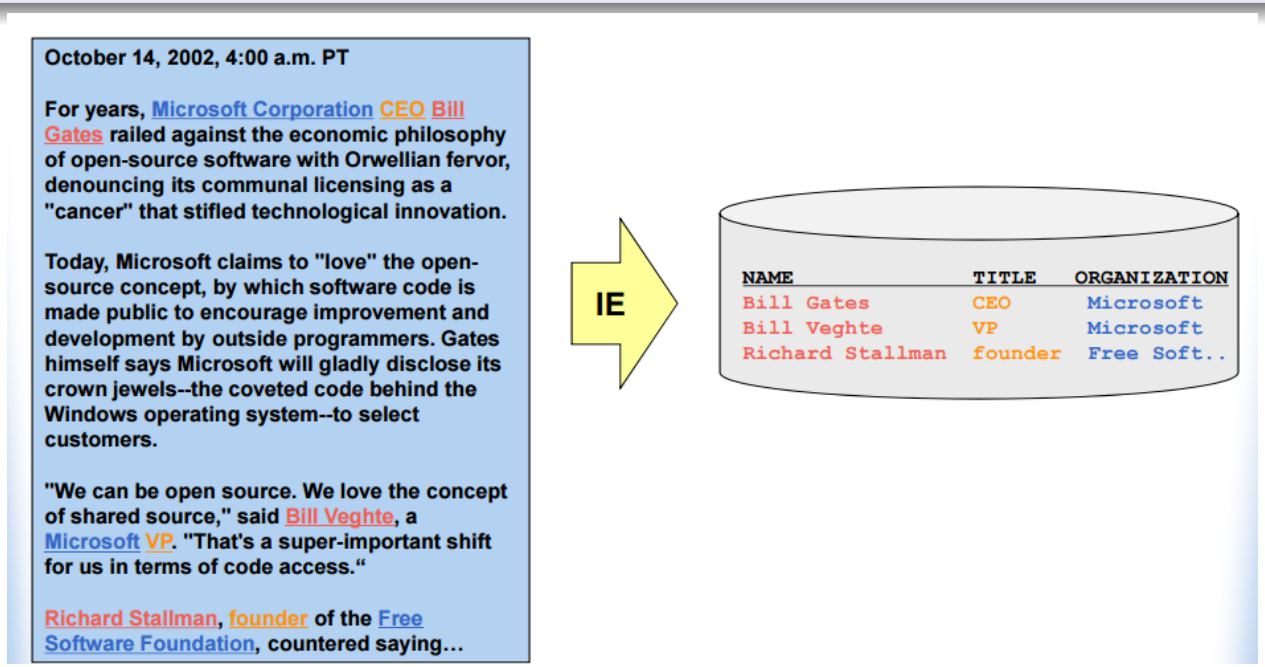
Named Entity Recognition

Named-entity recognition (NER) is a subtask of information extraction that seeks to locate and classify named entities in text into pre-defined categories such as the names of persons, organizations, locations, etc.

CHICAGO (AP) — Citing high fuel prices, United Airlines said Friday it has increased fares by $6 per round trip on flights to some cities also served by lower-cost carriers. American Airlines, a unit AMR, immediately matched the move, spokesman Tim Wagner said. United, a unit of UAL, said the increase took effect Thursday night and applies to most routes where it competes against discount carriers, such as Chicago to Dallas and Atlanta and Denver to San Francisco, Los Angeles and New York.

Information Extraction

Information extraction (IE) is the task of automatically extracting structured information from unstructured and/or semi-structured machine-readable documents



What makes natural language processing difficult?

1. Ambiguity
2. Language is organized in a hierarchical manner. Characters form words which form sentences which form documents which form ideas.
3. Order and context are extremely important. "I shot a photo" and "I shot a person" have vastly different meanings even though they differ by a very small amount.
4. The number of tokens is not fixed. A natural language can have hundreds of thousands of different words, new words are created on the fly, portmanteaus (e.g. stagflation) add to this complexity.
5. Languages are changing everyday, new words, new rules, etc.

Text representation

Tokenization

Given a character sequence and a defined document unit, tokenization is the task of chopping it up into pieces, called tokens , perhaps at the same time throwing away certain characters, such as punctuation. Here is an example of tokenization:

Input: Adrian, Sam, Bill want to sell fruits.
Output: ["Adrian", "Sam", "Bill", "want", "to", "sell", "fruits"]

Text representation

Stemming

Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word form. The stem need not be identical to the morphological root of the word.

Sample text: Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

Lovins stemmer: such an analys can reve featur that ar not eas vis from th vari in th individu gen and can lead to a pictur of expres that is mor biolog transpar and acces to interpres

Porter stemmer: such an analysi can reveal featur that ar not easili visibl from the variat in the individu gene and can lead to a pictur of express that is more biolog transpar and access to interpret

Paice stemmer: such an analys can rev feat that are not easy vis from the vary in the individ gen and can lead to a pict of express that is mor biolog transp and access to interpret

Text representation

Bag of words

- The bag of words model is a common way to represent documents in matrix form.
- We construct an nxt document-term matrix, where n is the number of documents, and t is the number of unique terms.
- Each column represents a unique term, and each cell i,j represents how many of term j are in document i.

Text representation

Bag of words

Example

1. I like solving interesting problems.
2. What is machine learning?
3. I'm not sure.
4. Machine learning predicts everything.

	eveyrthing	interesting	learning	lerning	like	Machien	machine	not	predicts	problems	solving	sure	What
1	0	1	0	0	1	0	0	0	0	1	1	0	0
2	0	0	1	0	0	0	1	0	0	0	0	0	1
3	0	0	0	0	0	0	0	1	0	0	0	1	0
4	1	0	0	1	0	1	0	0	1	0	0	0	0

Text representation

N-gram model

Example:
"John likes to watch movies. Mary likes movies too"
[
"John likes",
"likes to",
"to watch",
"watch movies",
"Mary likes",
"likes movies",
"movies too",
]

Text representation

TF-idf

In a large text corpus, some words will be very present (e.g. “the”, “a”, “is” in English) hence carrying very little meaningful information about the actual contents of the document.
Tf means **term-frequency** while tf–idf means term-frequency times **inverse document-frequency**:

$$tf-idf(t,d) = tf(t,d) \times idf(t)$$

Term frequency tf(t,d) is the raw frequency of a term in a document. The **inverse document frequency** is a measure of how much information the word provides, that is, whether the term is common or rare across all documents.

Text representation

Word2Vec

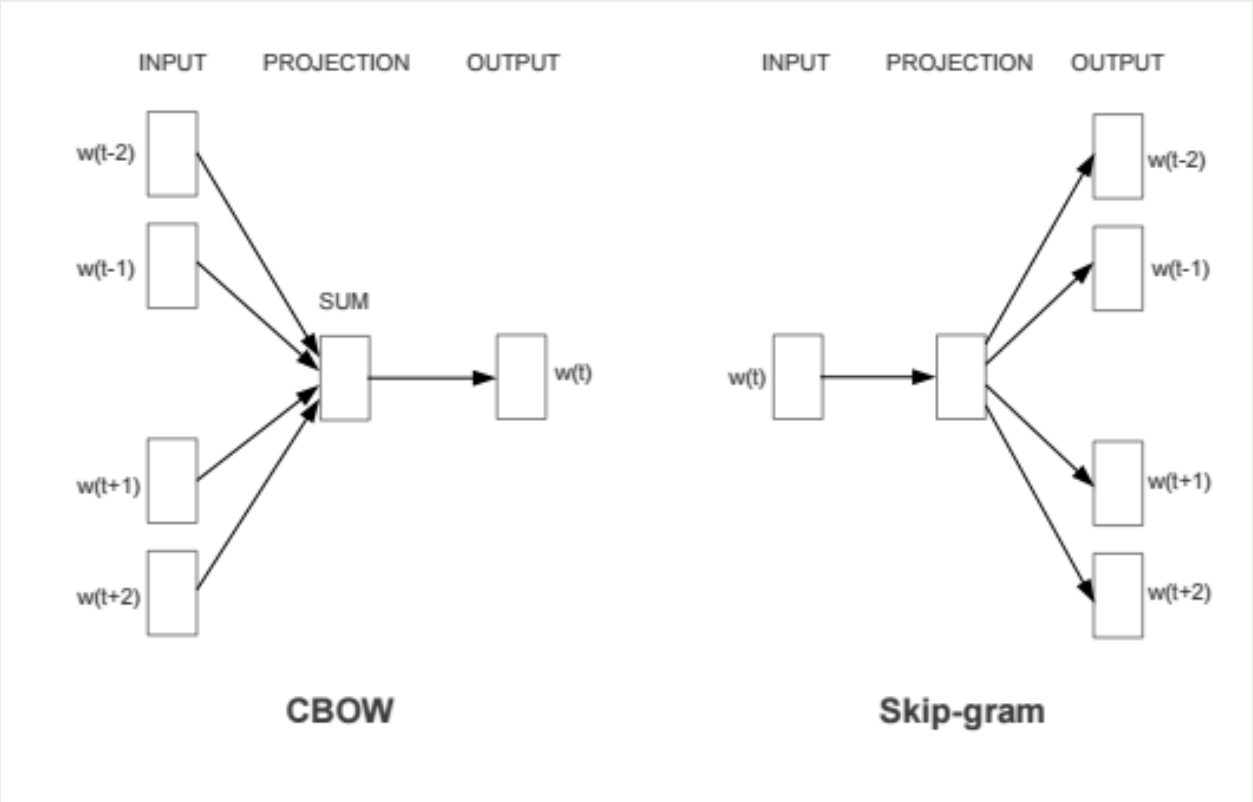
Word2vec is a particularly computationally-efficient predictive model for learning word embeddings from raw text.

- Lower computational complexity than previous representations (LDA,NNLM).
- Words with similar meanings have similar vectors.
- It is needed very large dataset to train this model.

Text representation

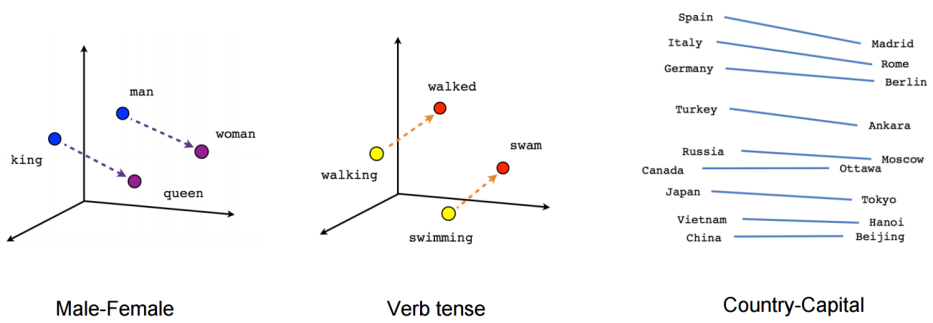
Word2Vec

Word2vec is not an algorithm it is framework.

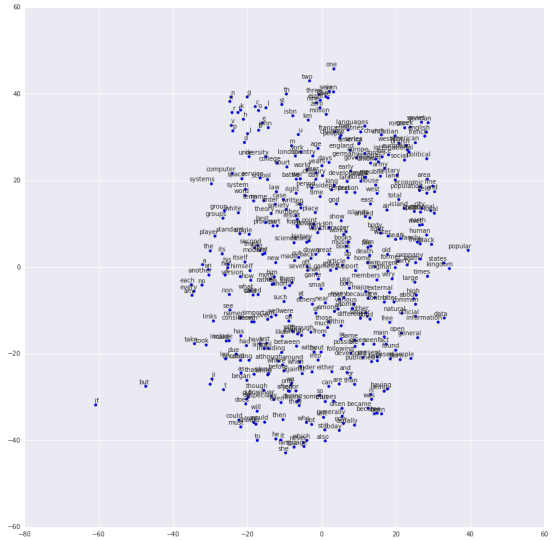


Text representation

Word2Vec visualization



(a) Vector properties



(b) TSNE visualization

Text representation

Another words embeddings

- GLOVE
- Skip-thought vectors

Recurrent Neural Networks

Another words embeddings

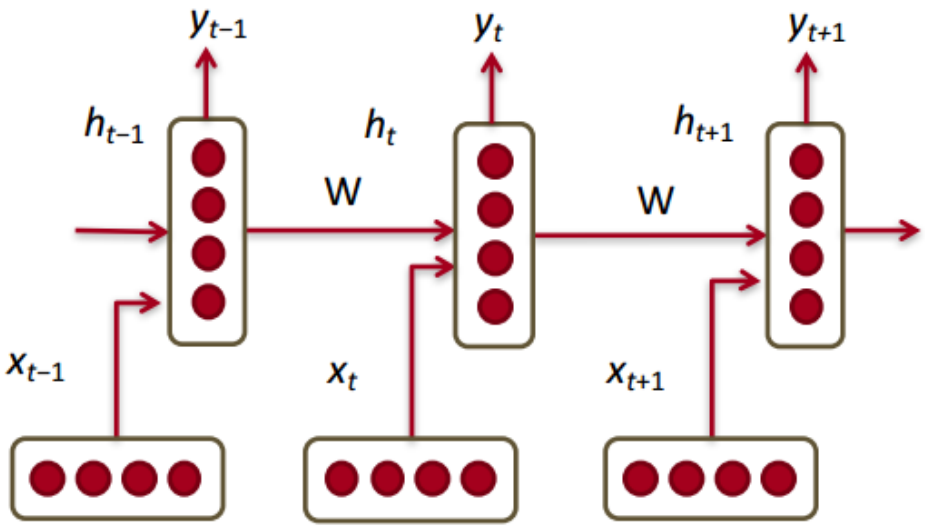
- GLOVE
- Skip-thought vectors

Traditional Language Models

- Performance improves with keeping around higher ngrams counts and doing smoothing and so-called backoff (e.g. if 4-gram not found, try 3-gram, etc)
- There are A LOT of n-grams! Gigantic RAM requirements!
- Recent state of the art: Scalable Modified Kneser-Ney Language Model Estimation by Heafield et al.: "Using one machine with 140 GB RAM for 2.8 days, we built an unprunedmodel on 126 billion tokens"

Recurrent Neural Networks

- RNNs tie the weights at each time step
- Condition the neural network on all previous words
- RAM requirement only scales with number of words



Recurrent Neural Networks

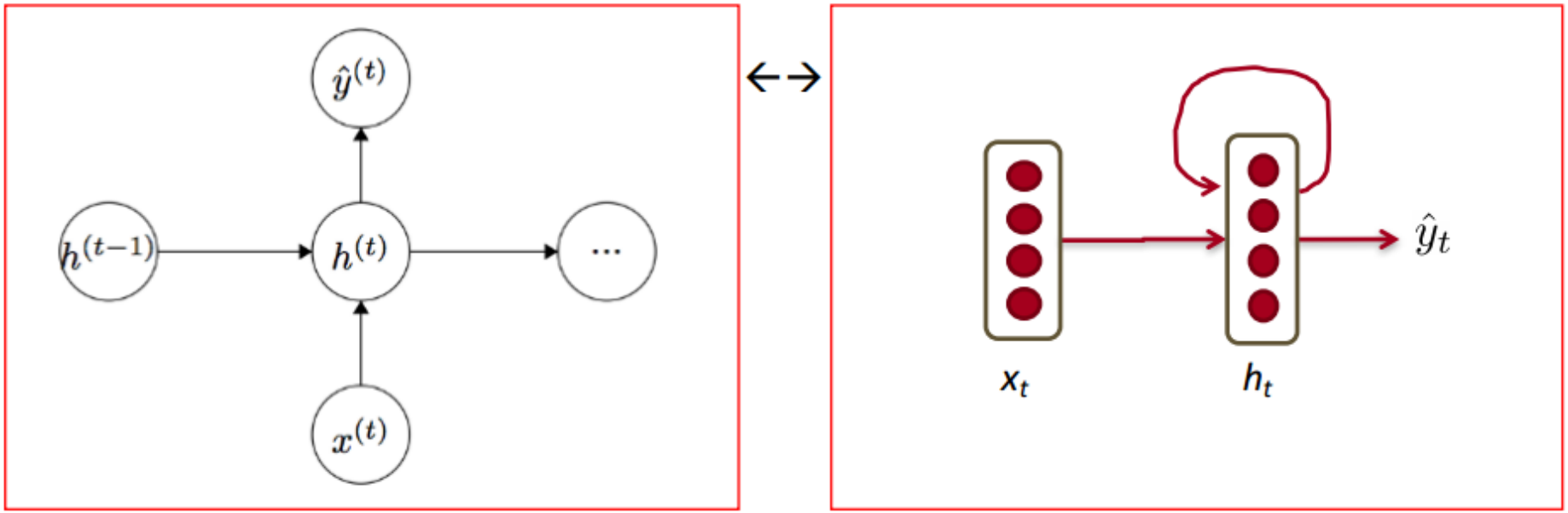
Given list of word **vectors:** $x_1, \ldots, x_{t-1}, x_t, x_{t+1}, \ldots, x_T$

At a single time step:

$$h_t = \sigma\left(W^{(hh)} h_{t-1} + W^{(hx)} x_{[t]}\right)$$

$$\hat{y}_t = \text{softmax}\left(W^{(S)} h_t\right)$$

$$\hat{P}(x_{t+1} = v_j \mid x_t, \ldots, x_1) = \hat{y}_{t,j}$$



Recurrent Neural Networks

Main idea: we use the same set of W weights at all time steps!

Everything else is the same:

$$
\begin{aligned}
h_t &= \sigma\left(W^{(hh)} h_{t-1} + W^{(hx)} x_{[t]}\right) \\
\hat{y}_t &= \text{softmax}\left(W^{(S)} h_t\right) \\
\hat{P}(x_{t+1} = v_j \mid x_t, \ldots, x_1) &= \hat{y}_{t,j}
\end{aligned}
$$

$h_0 \in \mathbb{R}^{D_h}$ is some initialization vector for the hidden layer at time step 0

$x_{[t]}$ is the column vector of L at index [t] at time step t

$W^{(hh)} \in \mathbb{R}^{D_h \times D_h}$ $W^{(hx)} \in \mathbb{R}^{D_h \times d}$ $W^{(S)} \in \mathbb{R}^{|V| \times D_h}$

Recurrent Neural Networks

$\hat{y} \in \mathbb{R}^{|V|}$ is a probability distribution over the vocabulary

Same cross entropy loss function but predicting words instead of classes

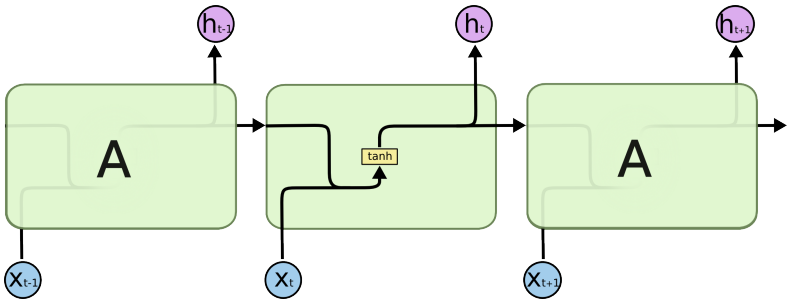
$$J^{(t)}(\theta) = -\sum_{j=1}^{|V|} y_{t,j} \log \hat{y}_{t,j}$$

Training RNNs is hard

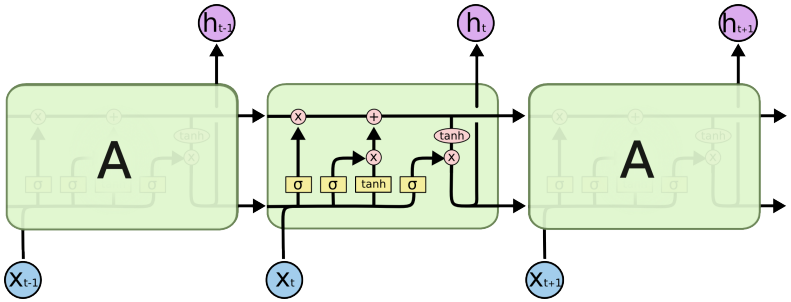
- Exploding and vanishing gradients
- Hard to handle long-term dependencies

LSTM Networks

Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN,



(c) The repeating module in a standard RNN contains a single layer.



(d) The repeating module in an LSTM contains four interacting layers.

Long-short-term-memories (LSTMs)

- We can make the units even more complex
- Allow each time step to modify
  - Input gate (current cell matters) $i_t = \sigma\left(W^{(i)}x_t + U^{(i)}h_{t-1}\right)$
  - Forget (gate 0, forget past) $f_t = \sigma\left(W^{(f)}x_t + U^{(f)}h_{t-1}\right)$
  - Output (how much cell is exposed) $o_t = \sigma\left(W^{(o)}x_t + U^{(o)}h_{t-1}\right)$
  - New memory cell $\tilde{c}_t = \tanh\left(W^{(c)}x_t + U^{(c)}h_{t-1}\right)$
- Final memory cell: $c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$
- Final hidden state: $h_t = o_t \circ \tanh(c_t)$

Long-short-term-memories (LSTMs)

- En vogue default model for most sequence labeling tasks
- Very powerful, especially when stacked and made even deeper (each hidden layer is already computed by a deep internal network)
- Most useful if you have lots and lots of data

Training RNNs is hard

- Recurrent Neural Networks are powerful
- A lot of ongoing work right now
- LSTMs even better

Machine translation

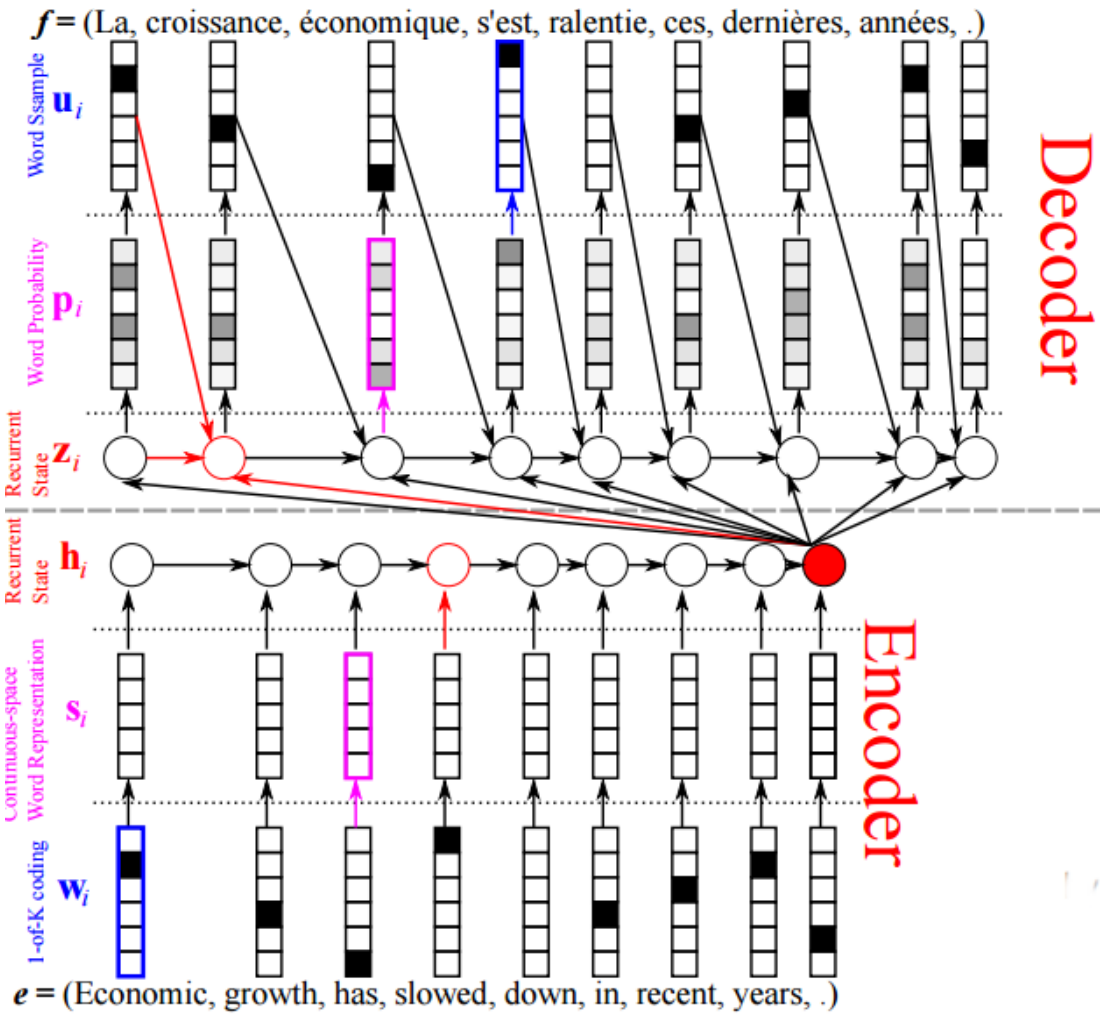
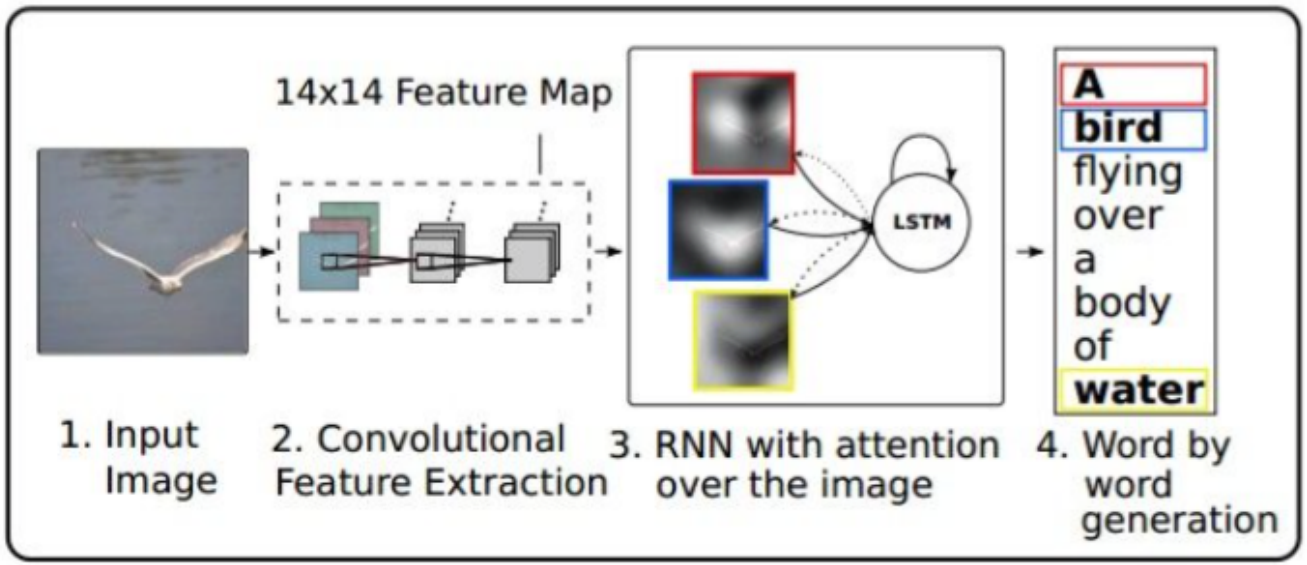
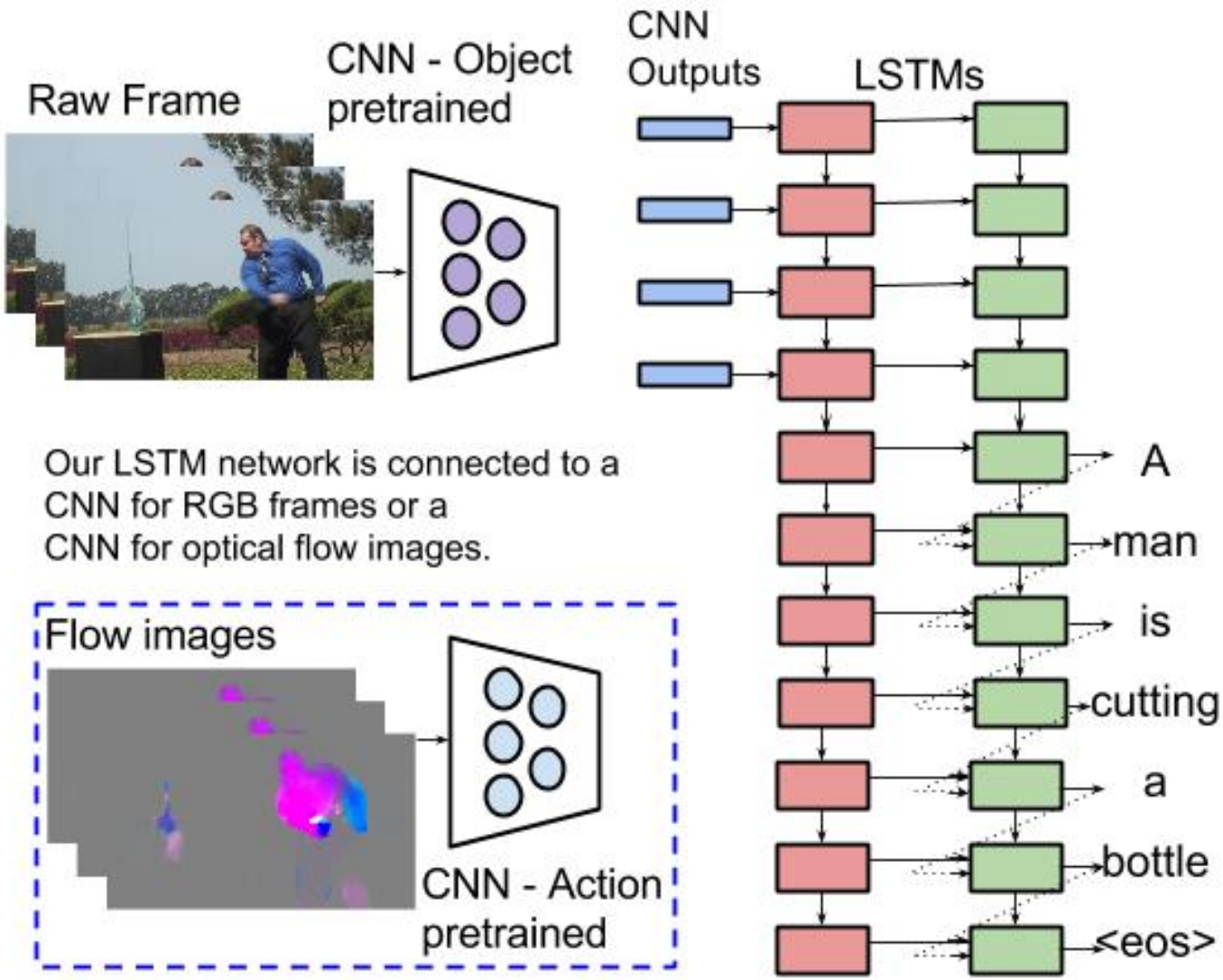


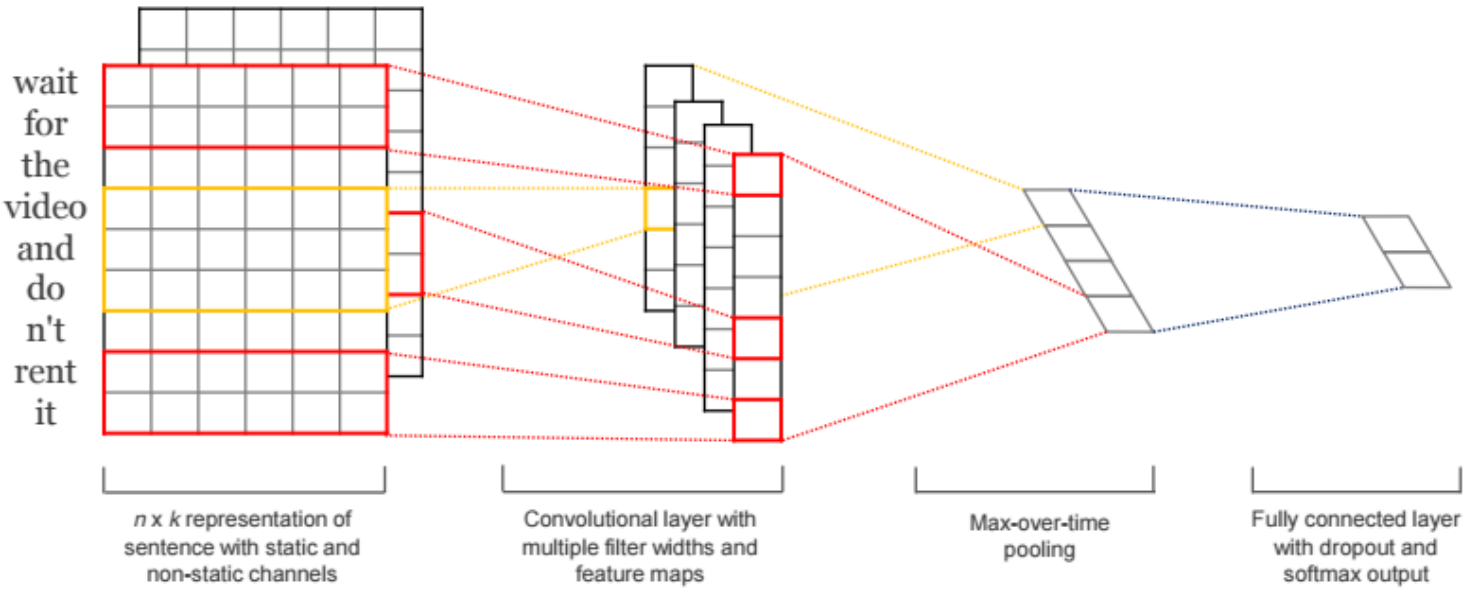
Image captioning



Video to text



Convolutional Neural Networks (for NLP)



Model comparison

- **Bag of Vectors**: Surprisingly good baseline for simple classification problems. Especially if followed by a few layers!
- **CNNs**: Good for classification, unclear how to incorporate phrase level annotation (can only take a single label), hard to interpret, easy to parallelize on GPUs
- **Recurrent Neural Networks**: Most cognitively plausible (reading from left to right), not usually the highest classification performance but lots of improvements right now with gates (GRUs, LSTMs, etc).
- Best but also most complex models: Hierarchical recurrent neural networks with attention mechanisms and additional memory