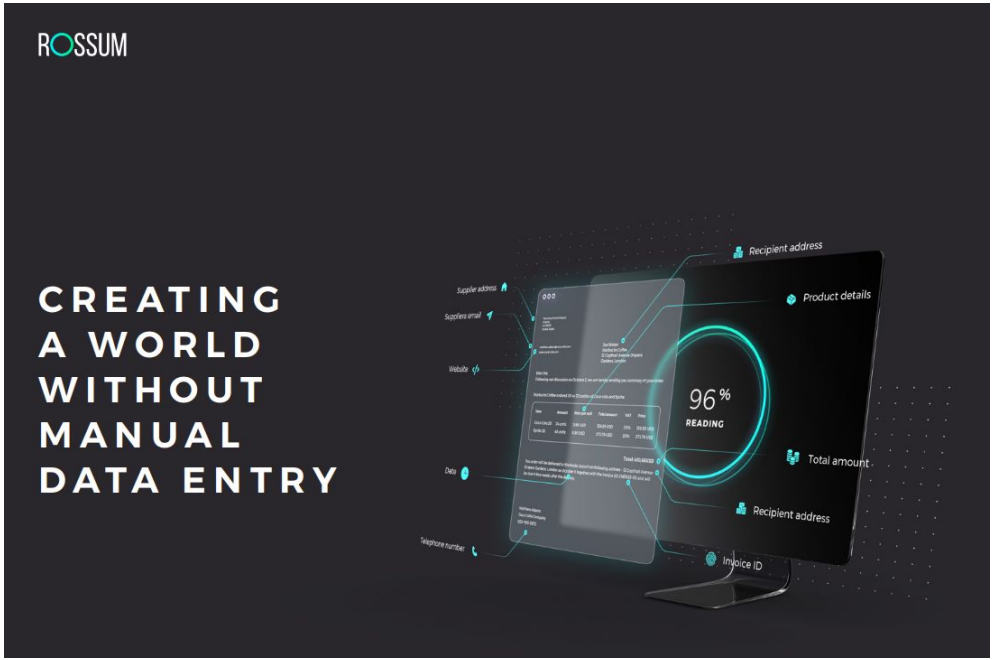
Deep Learning for Text Processing

Petr Baudis, Martin Holecek

ROSSUM

What we do @ ROSSUM



Building **Artificial Intelligence to understand documents**

Unique neural network architecture for **information extraction** from documents where layout is important, inspired by computer vision

In our Prague lab, we focus on **original research in deep learning** that is quickly applied in a real-world product

What will we do today

Introduction to Deep Learning techniques used in Natural Language Processing

We will introduce **neural networks** from scratch. Then, our first experiment will be an “understanding” task:

Sentiment classification of movie reviews

movie review text -> sentiment class (good vs. bad continuum)

Next, we will talk about **autoencoders**, a special neural network architecture. We will apply autoencoders to text understanding and use that as an example of how to analyze a model’s performance.

ROSSUM

Machine Learning on Text

Supervised Machine Learning task:

Dataset – a set of data samples

Data sample – (input, output) tuples

Input is mathematical text representation

Output is the to-be-predicted value(s)

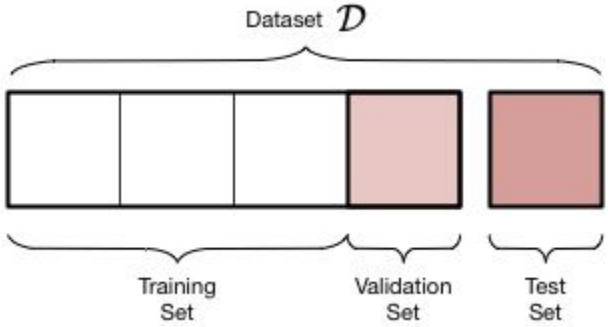
Model – neural network of a given architecture

f[θ](input) → output, where θ are trainable parameters

ROSSUM

Organization of data for machine learning

Splitting into a training part and a validation part needed to check how well our model generalizes on unseen examples



ROSSUM

How is text represented

ROSSUM

Text is a sequence

An item in a text sequence can be either a **character** or a **token** (~word)

Very similar to other temporal problems that involve sequences

- **Classification:** anomaly detection; **Prediction**: time series forecasting
- **Sequence generation:** music, handwriting, image captioning
- **Sequence to sequence:** translation, multi-step time series forecasting

Tokenization

Input: Friends, Romans, Countrymen, lend me your ears;
Output: [Friends] [Romans] [Countrymen] [lend] [me] [your] [ears]

Turning a raw text into a sequence of tokens by chopping it down and performing cleaning
Tokens can be words, sub-word units, punctuation...

Many ways to tokenize sentences:

Mr. O'Neill thinks that the boys' stories about Chile's capital aren't amusing.

For *O'Neill*, which of the following is the desired tokenization?

- [neill]
- [oneill]
- [o'neill]
- [o'] [neill]
- [o] [neill] ?

And for *aren't*, is it:

- [aren't]
- [arent]
- [are] [n't]
- [aren] [t] ?

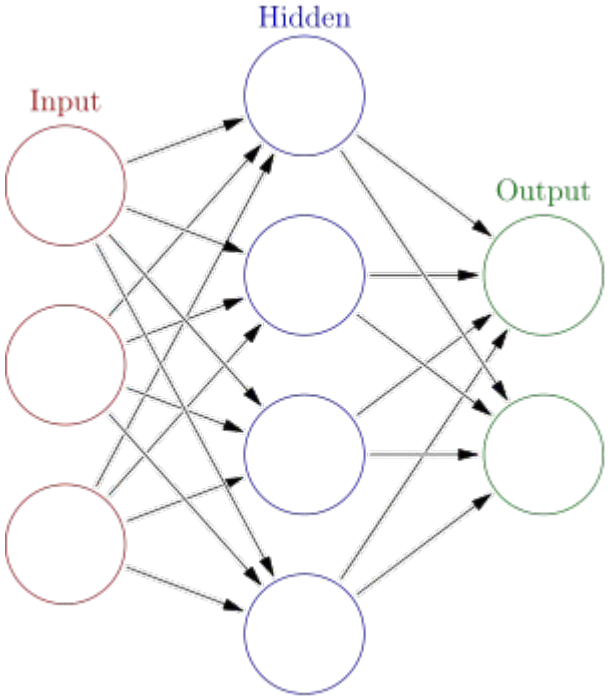
ROSSUM

Deep Learning

Training complicated Neural Networks with millions of parameters

Neural Networks are basically just mathematical functions usually with some real-valued inputs and outputs.

More details to come later today!



ROSSUM

Let’s get practical

Jupyter Notebook... in the cloud!

Google Colab - https://colab.research.google.com/

- Sign in using your google account (and cancel the main menu)
- **IMDb Task 1 Python notebook: http://tinyurl.com/mlprague19-imdb**
- **Autoencoder Python notebook: https://goo.gl/ehRE2U**
- Copy it to your google colab (File->Save a copy to drive)
- Feel free to try the accelerated runtime (Runtime->Change runtime type)

(This presentation: http://tinyurl.com/mlprague19-slides)

Wifi password: On your badge!

ROSSUM

Task 1: IMDb Reviews Sentiment

ROSSUM

this is one amazing movie!!!!! i myself did not understand everything but knowing chinese folklore (i studied them in school)it is very complicated. you just have to take what it gives you.....ENJOY THE MOVIE AND ENJOY THE RIDE....HOORAY!!!!

I hope the makers of this crap have day jobs because this film sucked!!! It looks like someones home movie and I don't think more than $100 was spent making it!!! Total crap!!! Who let's this stuff be released?!?!?!

Task 1: IMDb Reviews Sentiment

Goal: Predict sentiment from text input

We have labelled examples (dataset - supervised training).

Labels are 1/0 (good/bad) - categorical.

Machine Learning Model: Propose a mathematical formula that computes the sentiment from input.

Machine Learning Training: Find coefficients in the mathematical formula automatically.

How to encode text input mathematically?

ROSSUM

Mathematical Text Representation

Represent **tokens** *(or individual characters)* as:

- One-hot-encoded
- Vector embeddings

Represent **text** as:

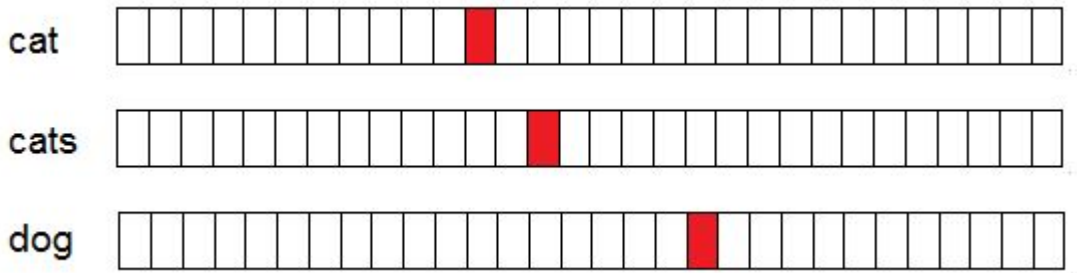
- Sequence of words
- Overall aggregation: “Bag-of-words”

ROSSUM

Token Representation

One-hot vector: all zeros except a single 1 at the index of the word

Unique vector for each word in our dictionary (vector length = vocabulary size)



ROSSUM

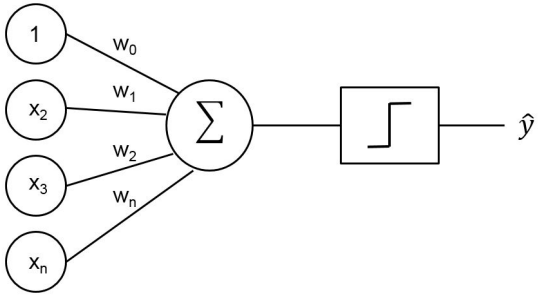
Aggregate Text Representation

Bag-of-Words: similar to one-hot vectors only now we indicate all words present in a piece of text, so multiple words can be ‘hot’ at the same time.

Variant: “**tf-idf**” encoding

ROSSUM

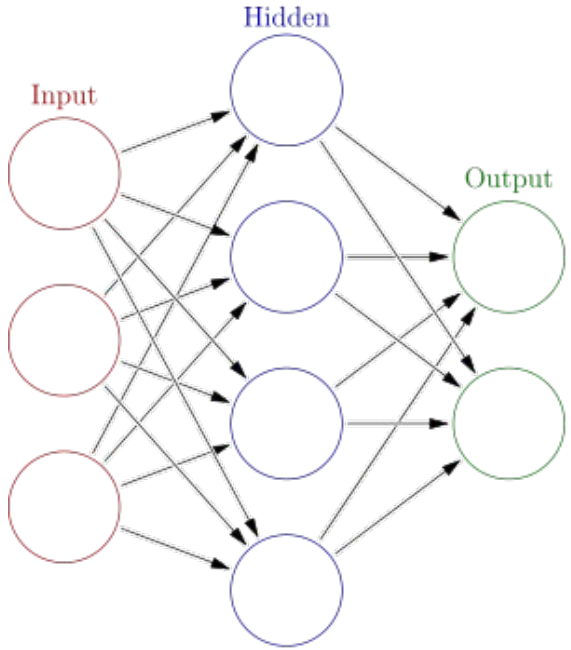
IMDb: First Experiment



Initial Idea: **Can we guess the sentiment based on isolated words?**

this is one **amazing** movie!!!!! i myself did not understand everything but knowing chinese folklore it is very *complicated*.

I *hope* the makers of this **crap** have day jobs because this film **sucked**!!!



Input: "bag-of-words" representation.
Model: Linear classifier.
(Maybe with an extra hidden layer?)

Model architectures for token sequences

Tokens: One-hot vectors become unwieldy (large vocabularies)

Vector Embeddings!

Sequence processing:

Aggregate – averaging, min/max, ...

Subsequences (“n-grams”) – convolutional CNNs

Memory – recurrent RNNs

Attention (we won’t cover this)

ROSSUM

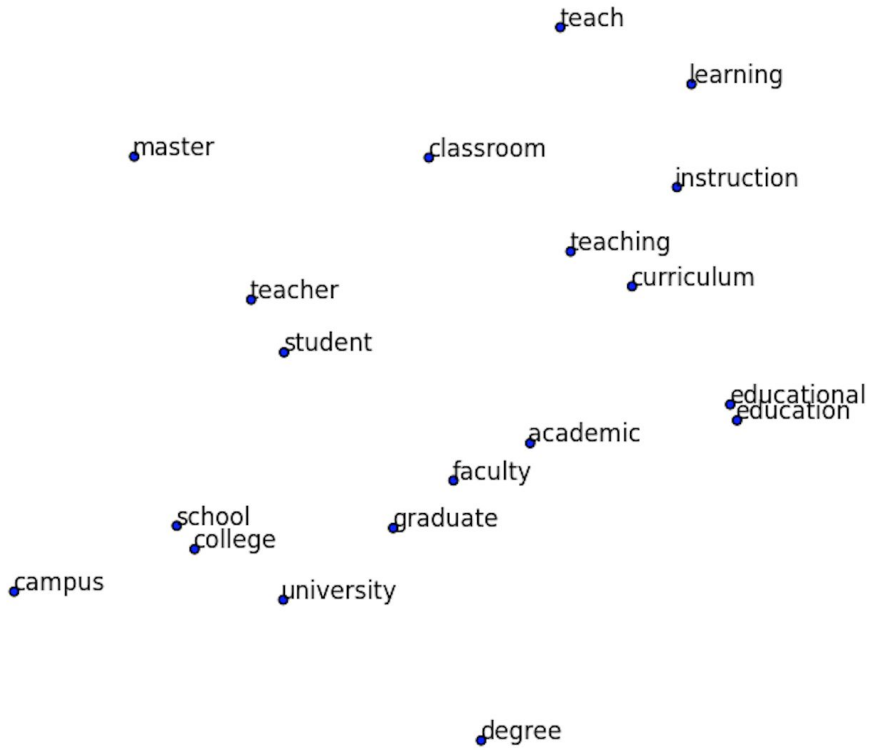
Token (Word) Representation

ROSSUM

Vector embedding: a real-valued vector for each word or char in our dictionary

Word embeddings are usually pre-trained on huge texts encoding their context using fast algorithms (Word2Vec, GloVe) – **prior semantics**

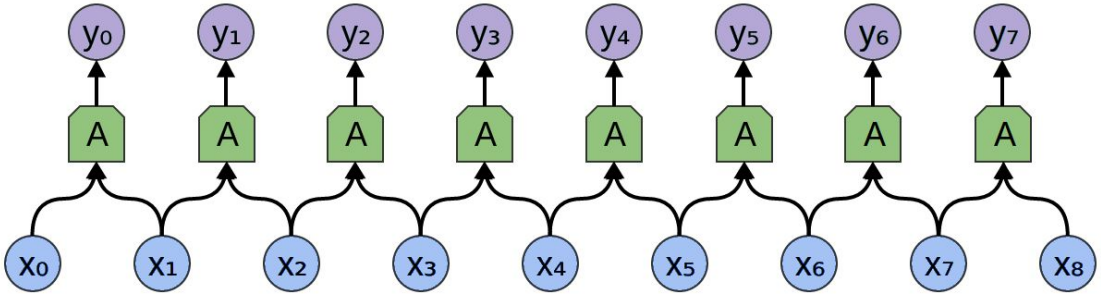
Similar words tend to have similar word embeddings; arithmetics often works



Neural Network architectures used with sequences

Convolutional: sliding window over a sequence that applies multiple *filters* on the window elements

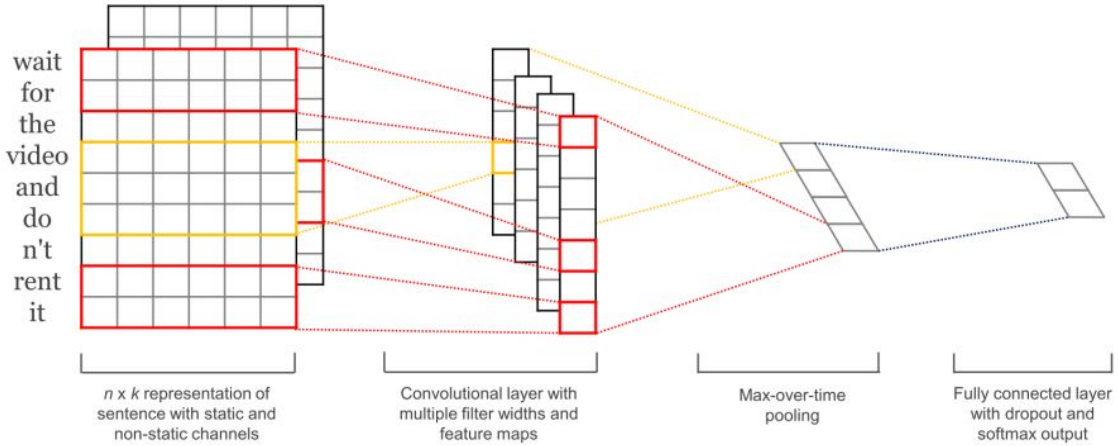
That means we can recognize and match multi-word patterns.



ROSSUM

Classifying the whole sentence or document

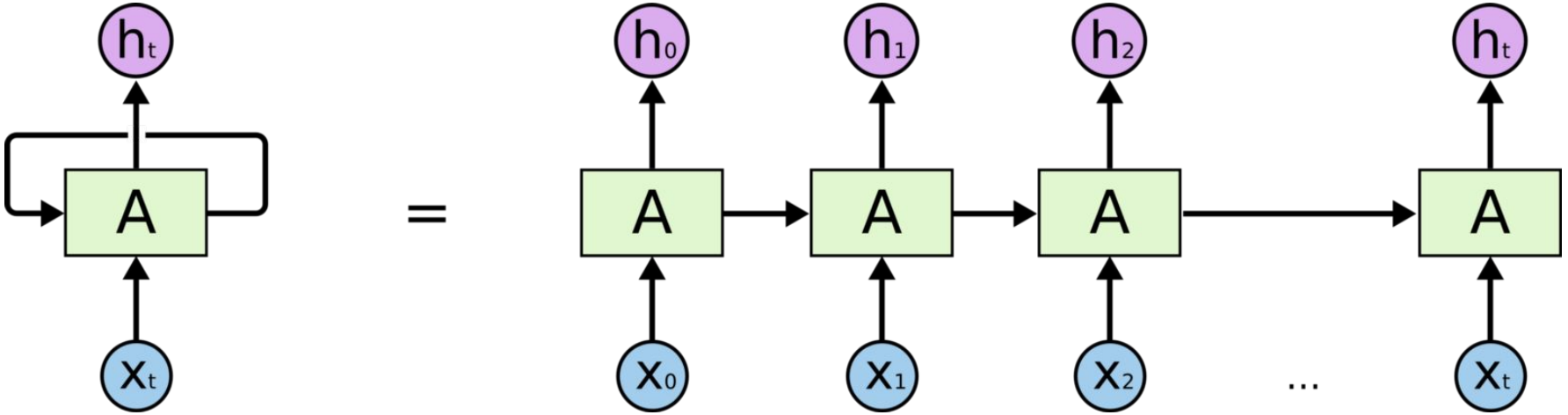
Convolutional: perform ‘Max Pooling’ (maximum over CNN output’s time axis)



ROSSUM

Neural Network architectures used with sequences

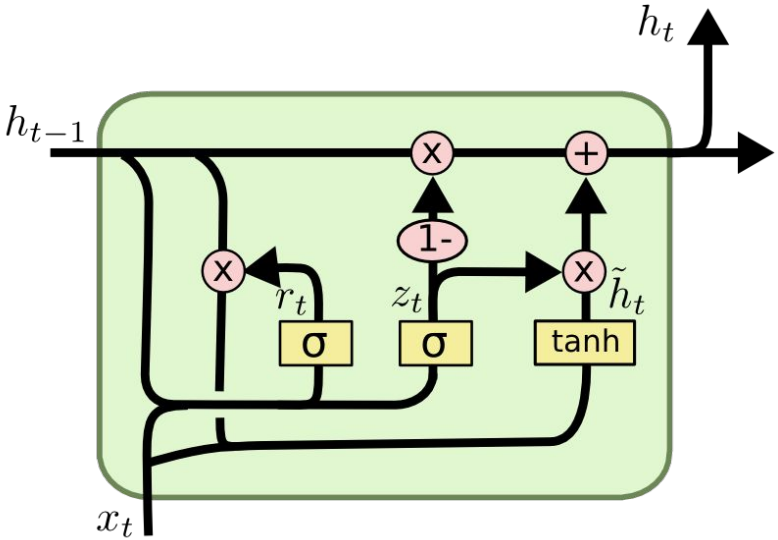
Recurrent: Analyze each individual element, but remember past analysis



ROSSUM

Neural Network architectures used with sequences

Recurrent: Gated Recurrent Unit (GRU); sequence of cells with **controlled memory mechanism**



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

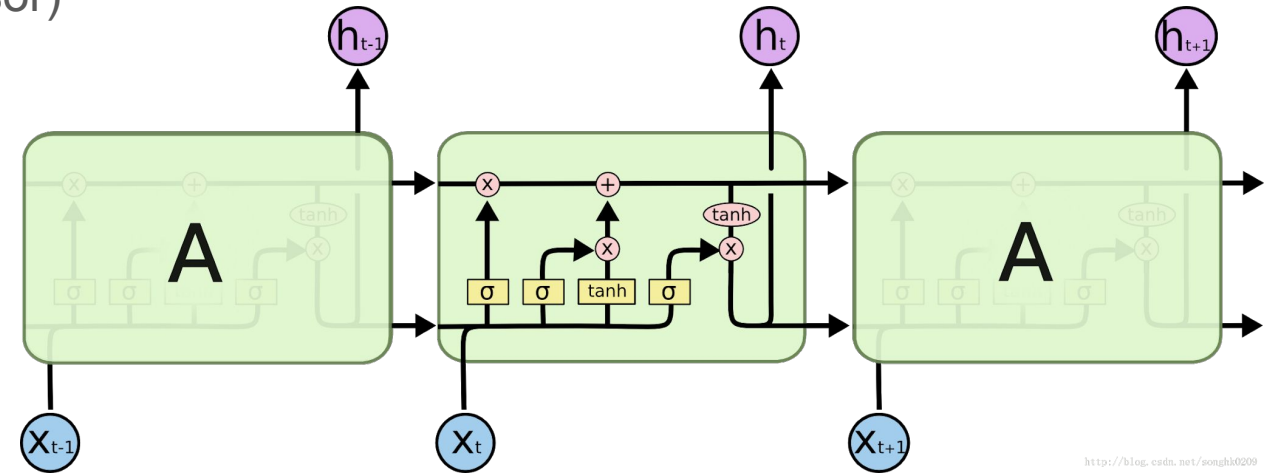
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Neural Network architectures used with sequences

Recurrent: Long Short Term Memory (LSTM); more complicated GRU variant (predecessor)



ROSSUM

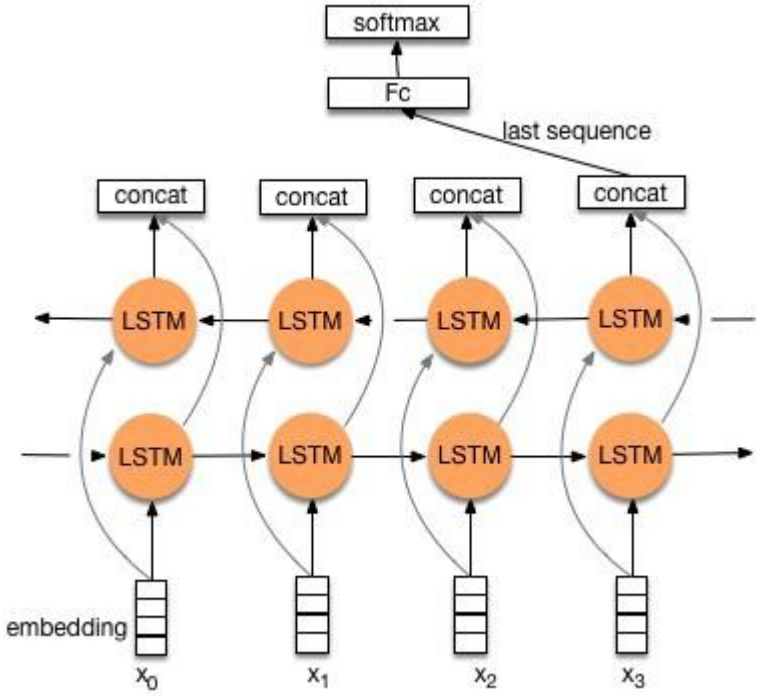
Neural Network architectures used with sequences

Recurrent: Bidirectional RNNs

Run forwards and backwards; concatenate

Allows us to look both into past and future

(LSTM or GRU, doesn't matter…)



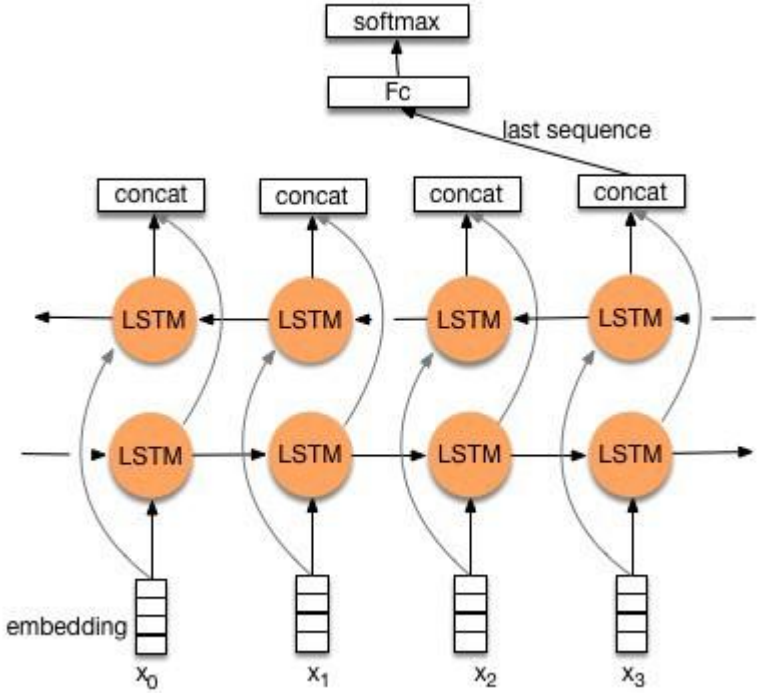
ROSSUM

Classifying the whole sentence or document

Recurrent:

simply take only the last state of the sequence

use as the *hidden state*, train a classifier based on this state



ROSSUM

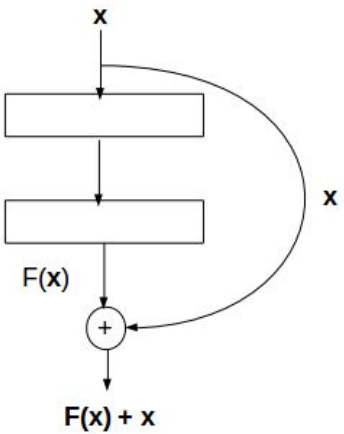
Lego Bricks of Deep Learning!

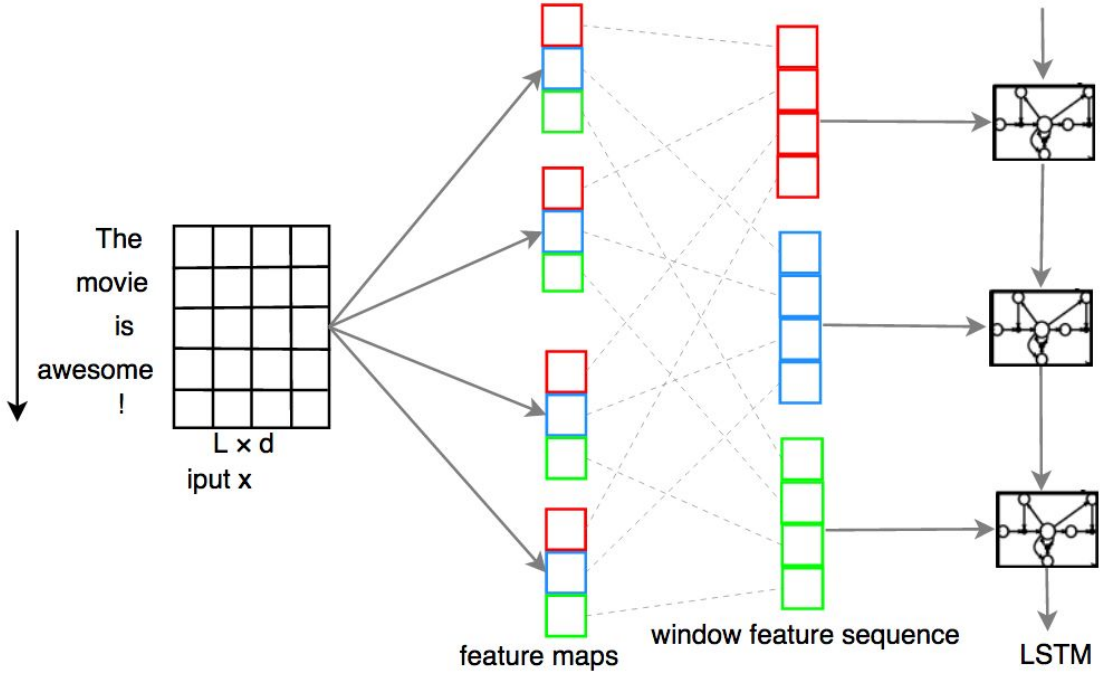
ROSSUM

CNN + RNN architecture

Shortcuts & inception

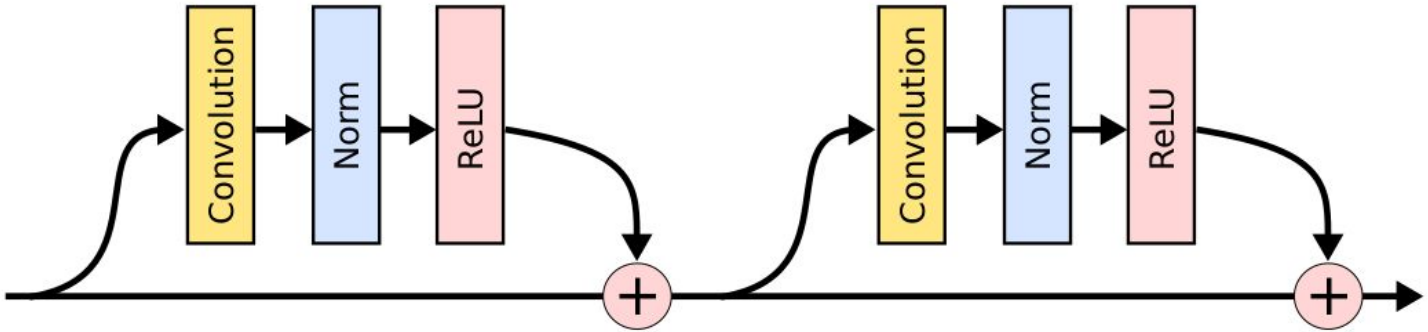
Ensembles





Lego Bricks of Deep Learning!

Shortcuts & inception



The Autoencoder Journey

Next, we will talk about **autoencoders**, a special neural network architecture. We will apply autoencoders to text understanding - **not to show astonishing results** but to talk about **how to analyze a model's performance**.

- **Autoencoder Python notebook: https://goo.gl/ehRE2U**
- Copy it to your google colab (File->Save a copy to drive)
- Feel free to try the accelerated runtime (Runtime->Change runtime type)

(This presentation: http://tinyurl.com/mlprague19-slides)

ROSSUM

The Autoencoder Key Takeaways

Autoencoders are powerful and have high learning capacity

...but that may not mean the model will do anything useful!

Task is king - focus on performance in your target task

Focus on internal organization

(ex.: position-specific encoding is often a no-no)

Anomalies and clustering can help indicate glitches

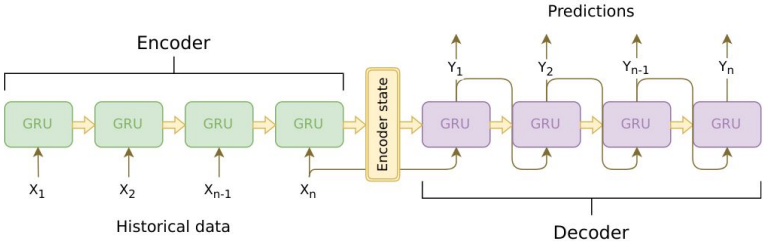
ROSSUM

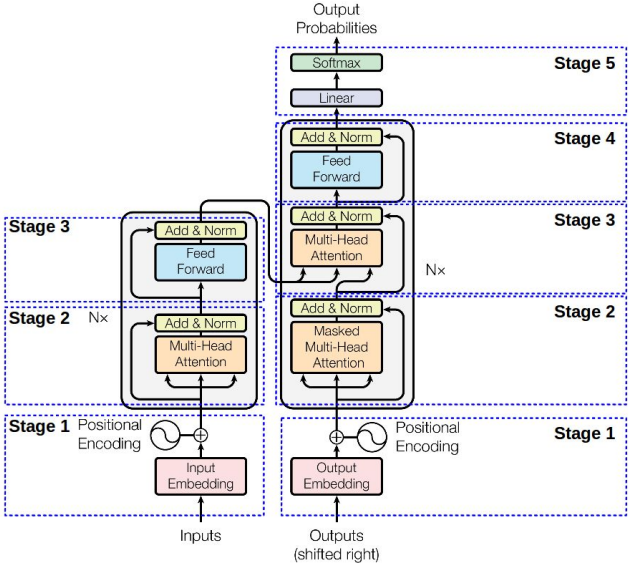
Best Autoencoders for Text?

“seq2seq” tasks!

What about just an RNN? (LSTM, GRU)

Best now: **Transformers**.
(Best transformer: GPT2 by OpenAI.)





Conclusion

Deep learning is no (complicated) magic!
Even simple approaches (bag of words) can work great. Just represent data well.

Don’t be afraid to put together DL components as lego bricks.
Most of details in your architecture decisions just won’t matter.

This presentation: https://bit.ly/2TXvZ4x

ROSSUM

References

http://colah.github.io

https://towardsdatascience.com/

https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

https://arxiv.org/abs/1603.06127 cites a lot of classic deep learning papers for sentence processing

ROSSUM