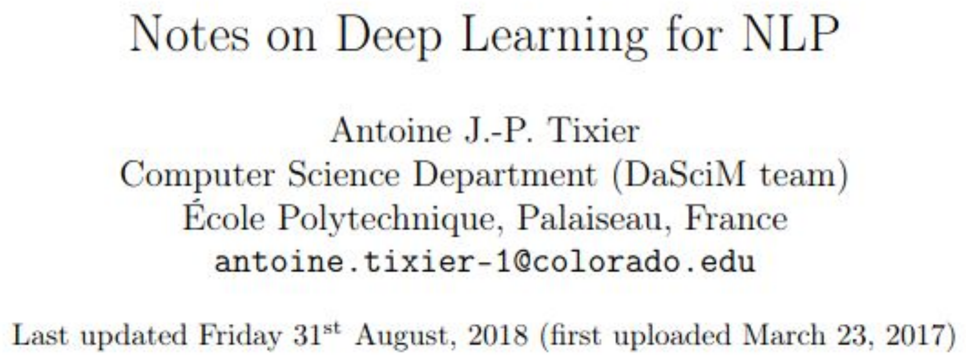
Notes on Deep Learning for NLP

Gabriel Lima Novais
Franklin Alves de Oliveira

The paper


Notes on Deep Learning for NLP

Antoine J.-P. Tixier
Computer Science Department (DaSciM team)
École Polytechnique, Palaiseau, France
antoine.tixier-1@colorado.edu

Last updated Friday 31st August, 2018 (first uploaded March 23, 2017)

link here.

Contents

NLP in a few slides…

What is NLP?

- Short for *Natural Language Processing*
- Not a new concept… (first patents date back to 1930)
- Is an interdisciplinary field concerned with the interactions between computers and human natural languages

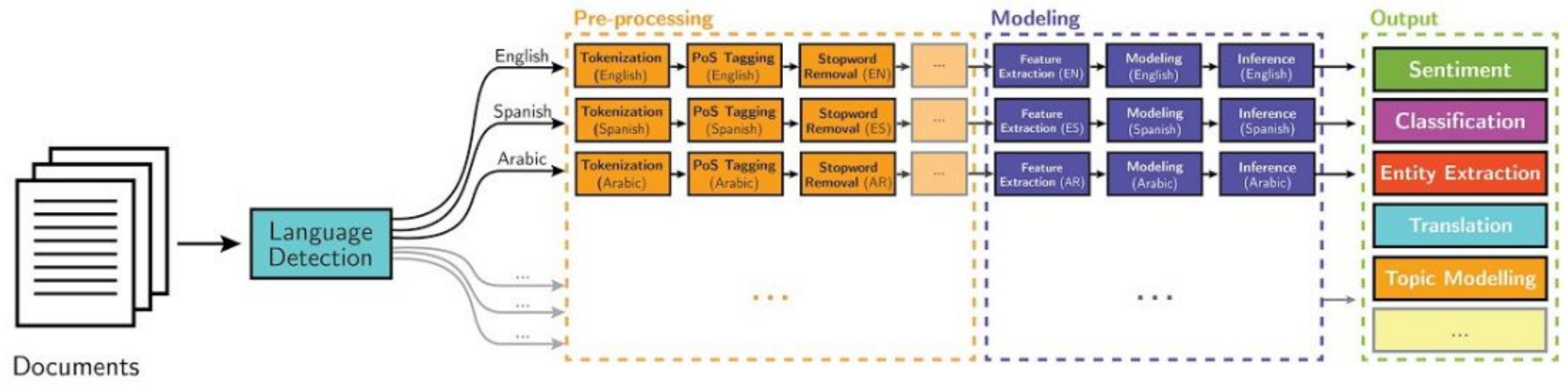
Okay, but why do we need this?

Interesting applications

- Sentiment Analysis
- Identify Fake News
- Stop spams
- Voice driven interfaces (like Siri, Alexa, and others)
- Machine Translation (ex.: Google Translate)
- Spell Checking
- ...

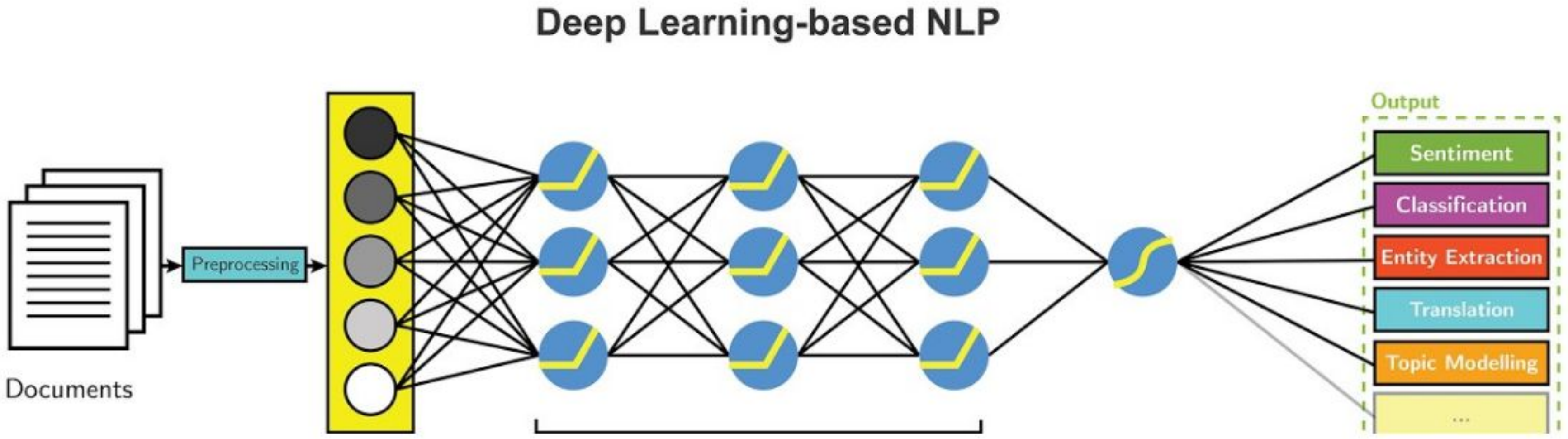
Sources: Towards Data Science (link-1 and link-2)

Pipeline



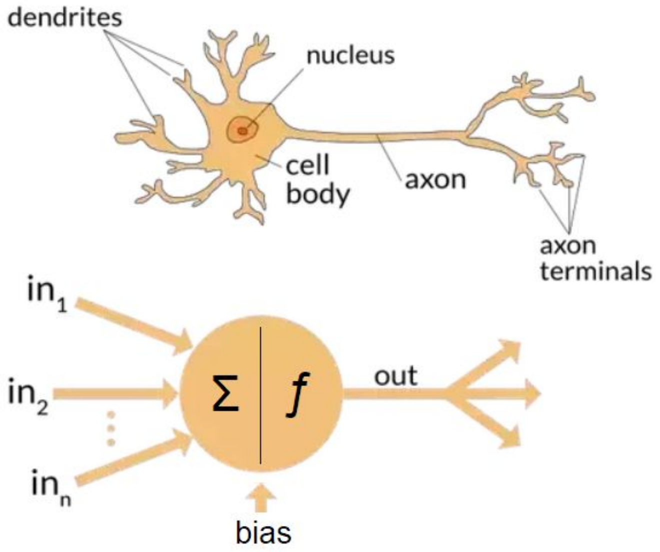
Documents
Language Detection
English
Spanish
Arabic
...
Pre-processing
Tokenization (English)
PoS Tagging (English)
Stopword Removal (EN)
Tokenization (Spanish)
PoS Tagging (Spanish)
Stopword Removal (ES)
Tokenization (Arabic)
PoS Tagging (Arabic)
Stopword Removal (AR)
Modeling
Feature Extraction (EN)
Modeling (English)
Inference (English)
Feature Extraction (ES)
Modeling (Spanish)
Inference (Spanish)
Feature Extraction (AR)
Modeling (Arabic)
Inference (Arabic)
Output
Sentiment
Classification
Entity Extraction
Translation
Topic Modelling
...

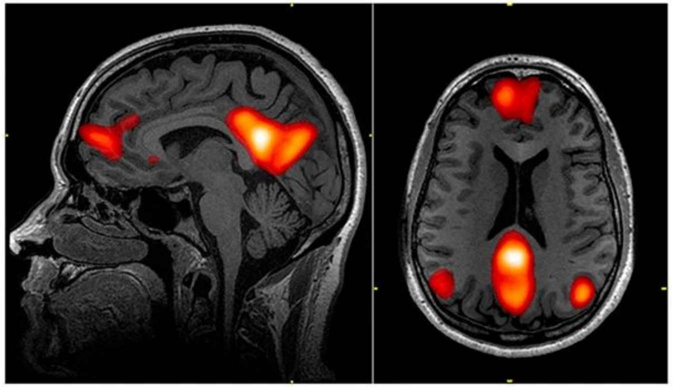
Pipeline with DNNs



Why using DNNs in NLP?

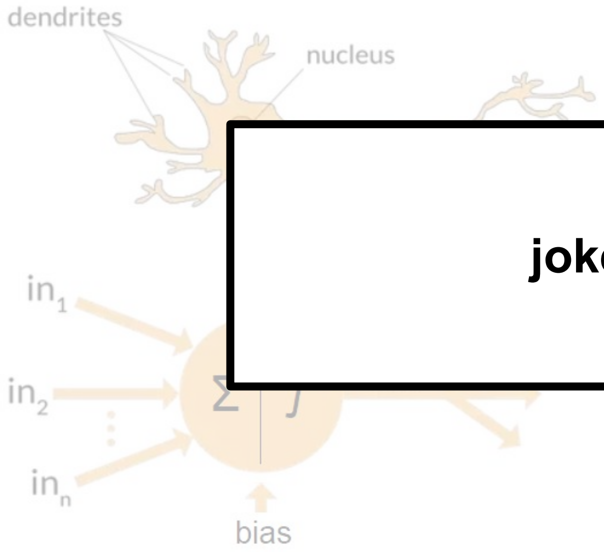
- Language is a human thing and DNNs were inspired by the human brain...

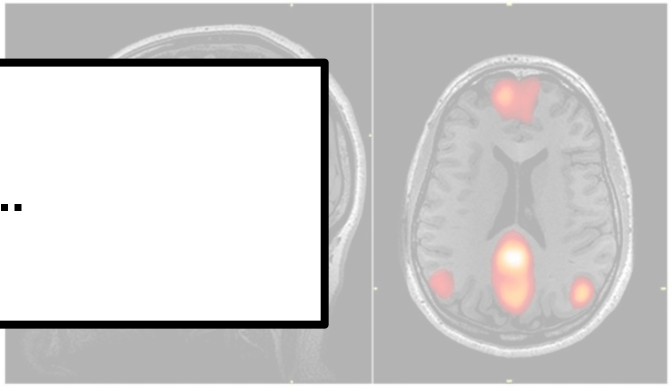




Why using DNNs in NLP?

- Language is a human thing and DNNs were inspired by the human brain...





jokes apart...

Why using DNNs in NLP?

- Deep Learning provides a very flexible, universal, and learnable framework for representing the world, for both visual and linguistic information.

- These models can often be trained with a single end-to-end structure and do not require traditional, task-specific feature engineering.

Read more on this link

Paradigm switch

Feature embeddings

- ***Deep Learning models*** often embed core features (and core features only) as vectors in a low-dimensional continuous space where dimensions represent *shared latent concepts*.
- The embeddings are initialized randomly or obtained from pre-training.
- They can then be updated during training just like other model parameters, or be kept static.

Benefits of feature embeddings

- The main advantage of mapping features to dense continuous vectors is the ability to capture similarity between features, and therefore to generalize.
- **for example:** “Obama” and “President”
    - With traditional one-hot vectors, those two features would be considered orthogonal and predictive power would not be able to be shared between them.
    - Also, going from a huge sparse space to a dense and compact space reduces computational cost and the amount of data required to fit the model, since there are fewer parameters to learn.

Combining core features

- Combinations of core features are not encoded as new dimensions of the feature space, but as the sum, average, or concatenation of the vectors of the core features that are to be combined.
- But some of these approaches completely ignore the ordering of the feature.

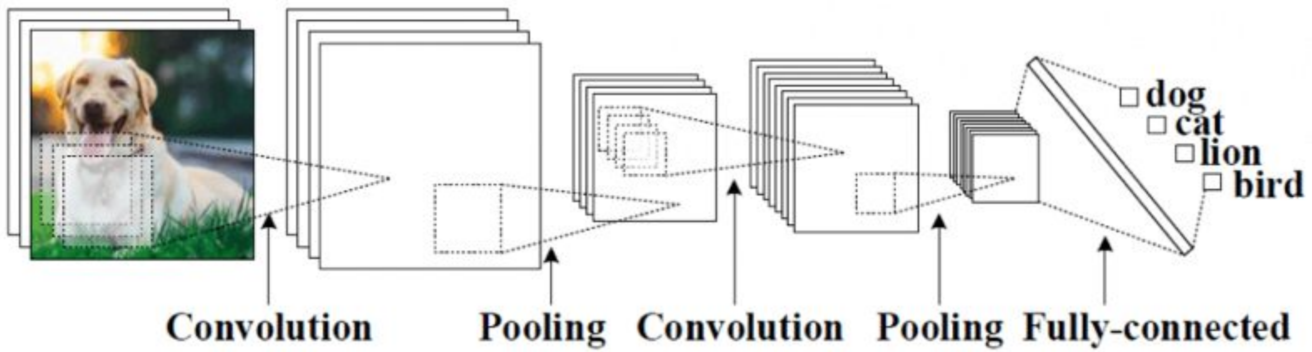
  “John is quicker than Mary” = ***“Mary is quicker than John”***
- On the other hand, using concatenation allows to keep track of ordering
- Every document in the collection can be transformed to have the same fixed length s:
  - ***Longer*** documents are truncated to their first *s* words
  - ***Shorter*** documents are padded with a special zero vector to make up for the missing word

Convolutional Neural Networks (CNNs)

What are CNNs?

- They are feedforward neural networks where each neuron in a layer receives input from a neighborhood of the neurons in the previous layer.

- CNNs were developed in computer vision to work on regular grids such as images



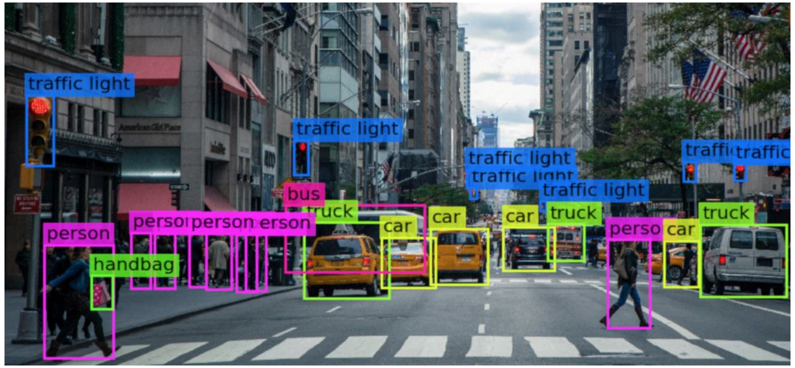
Key concepts:

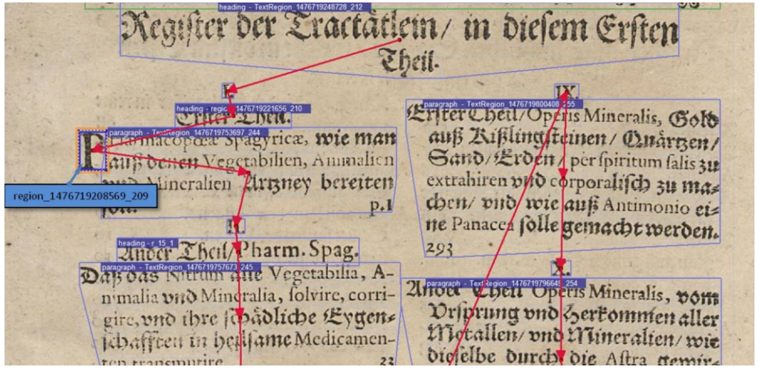
- Neighborhoods, or local receptive fields, allow CNNs to recognize more and more complex patterns in a hierarchical way, by combining lower-level, elementary features into higher-level features. This property is called ***compositionality***.

- The absolute positions of the features in the image do not matter. Only capturing their respective positions is useful for composing higher-level patterns. This property is called ***local invariance.***

Local invariance and compositionality

- CNNs have reached very good performance in computer vision, but it is not difficult to understand that thanks to ***compositionality*** and ***local invariance***, they can also do very well in NLP.




heading - TextRegion_1476719248728_212
heading - region_1476719221656_210
paragraph - TextRegion_1476719753697_244
region_1476719208569_209
heading - r_15_1
paragraph - TextRegion_1476719757673_245
paragraph - TextRegion_1476719800408_255
paragraph - TextRegion_1476719796649_254

Regiſter der Tractätlein/ in dieſem Erſten Theil.

I.

Erſter Theil.

Pharmacopœæ Spagyricæ, wie man auß denen Vegetabilien, Animalien und Mineralien Artzney bereiten ſoll. p.1

II.

Ander Theil/ Pharm. Spag.

Daß das Nitrum alle Vegetabilia, Animalia und Mineralia, ſolvire, corrigire, und ihre ſchädliche Eygenſchafften in heylſame Medicamenten transmutire. 23

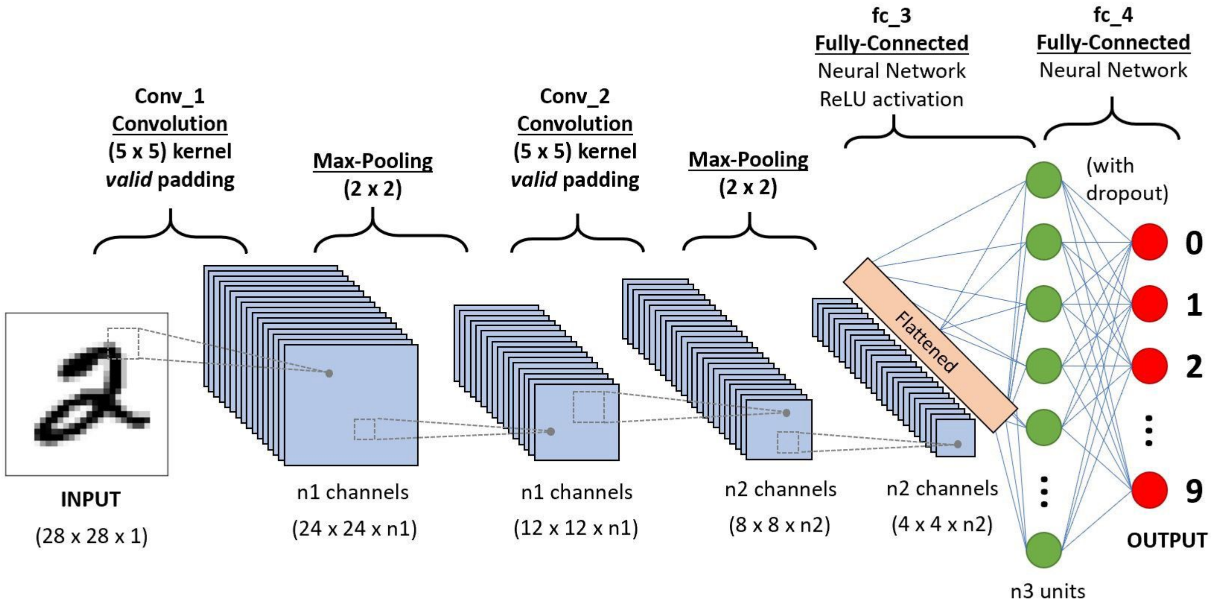
IX.

Erſter Theil/ Operis Mineralis, Gold auß Kißlingſteinen/ Quärtzen/ Sand/ Erden/ per ſpiritum ſalis zu extrahiren und corporaliſch zu machen/ und wie auß Antimonio eine Panacea ſolle gemacht werden. 293

X.

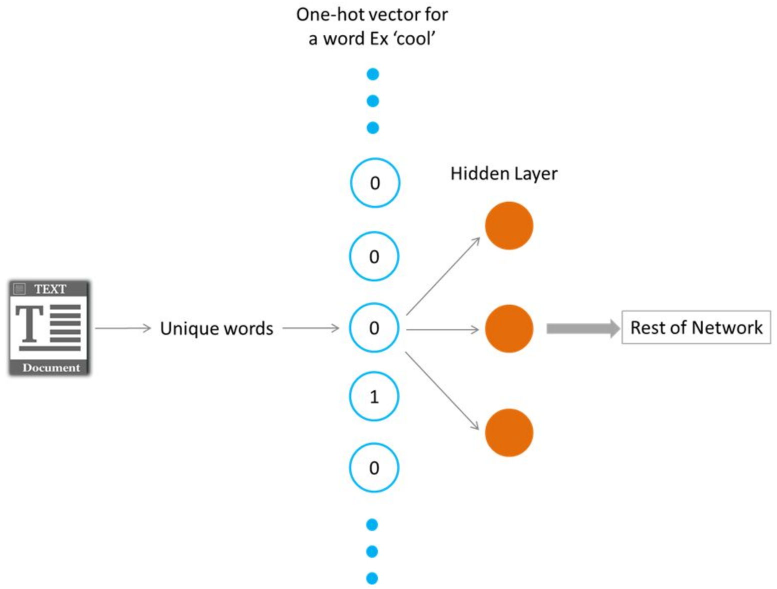
Ander Theil Operis Mineralis, vom Urſprung und Herkommen aller Metallen/ und Mineralien/ wie dieſelbe durch die Aſtra generi

Diving deeper into CNNs



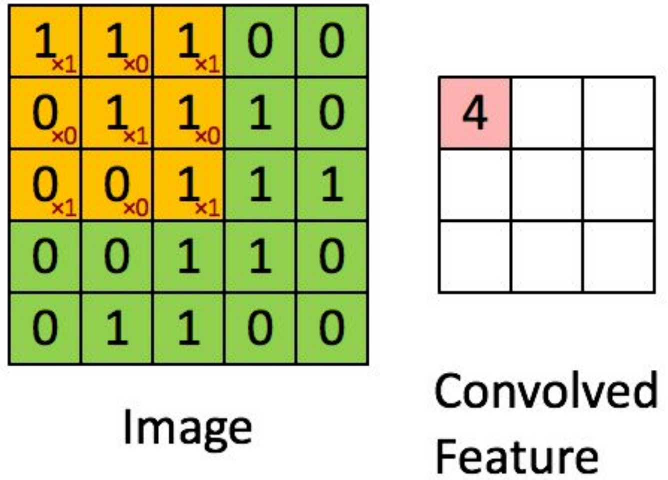
Input layer

- We can represent a document as a real matrix $A \in R^{s \times d}$, where s is the document length, and d is the dimension of the word embedding vectors



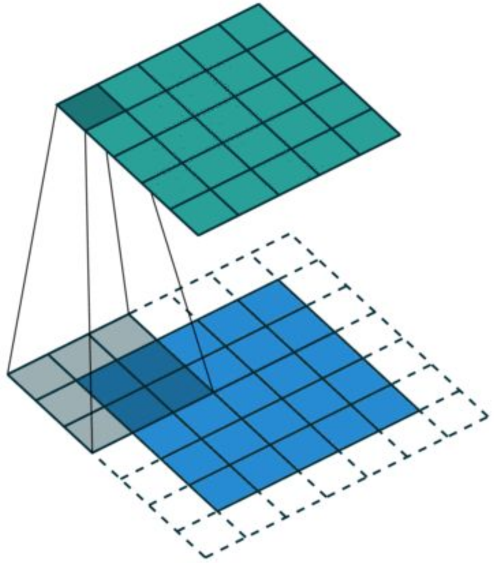
Convolution layer

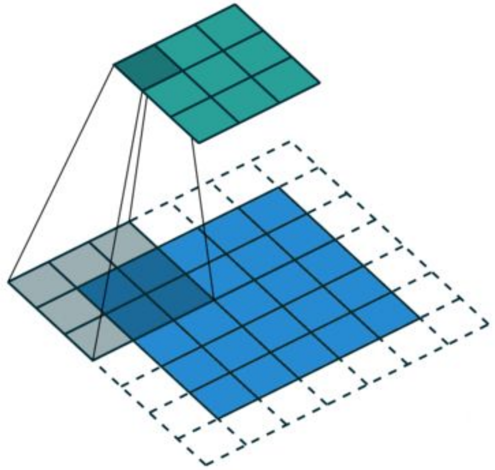
- The ***convolution layer*** is a linear operation followed by a nonlinear transformation.



Convolution layer

- ***Stride***: step size.


stride = 1


stride = 2

Convolution layer

- The instantiations of the window over the input are called ***regions*** or ***receptive fields***.

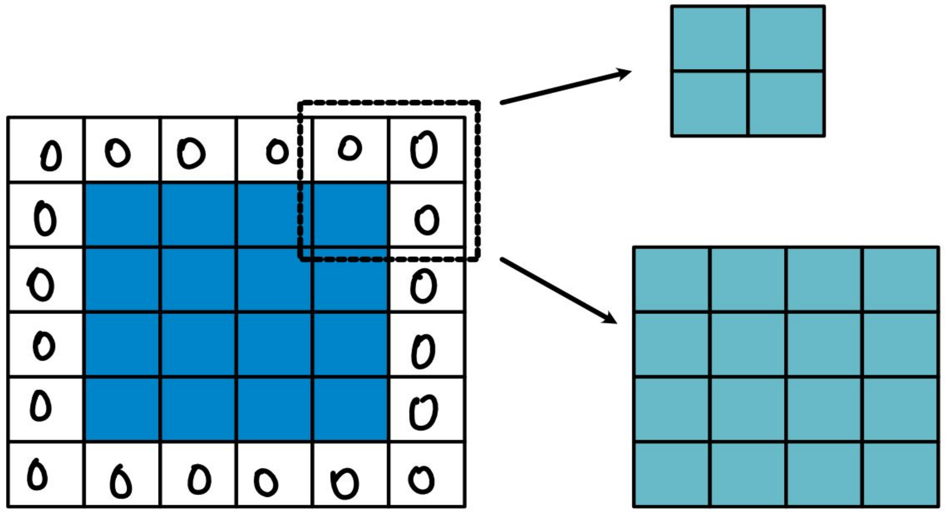
- There are (s−h)/stride + 1 of them, where stride corresponds to the number of words by which we slide the window at each step. (*h = window size; s = number of rows*)

- The output of the convolution layer for a given filter is thus a vector o ∈ R s−h+1 whose elements are computed as:

$$\sigma_i = W \cdot A[i : i + h - 1, :]$$

Convolution layer

- **Padding**: what to do when the filter reaches the end of the image.

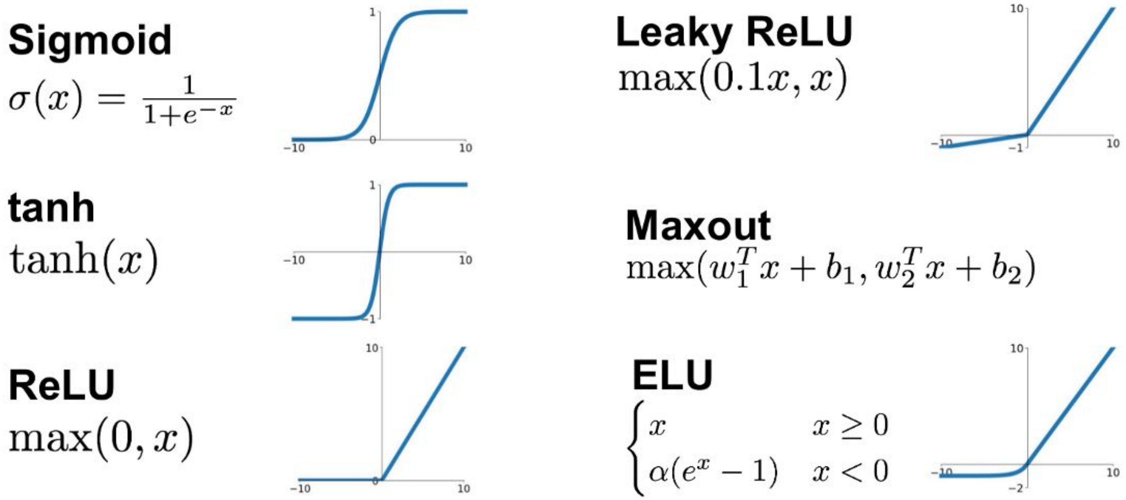


`padding = `**`valid`**

`padding = `**`same`**

Convolution layer

- Then, a nonlinear activation function f is applied elementwise to σ, returning what is known as the feature map c ∈ R s−h+1 associated with the filter:

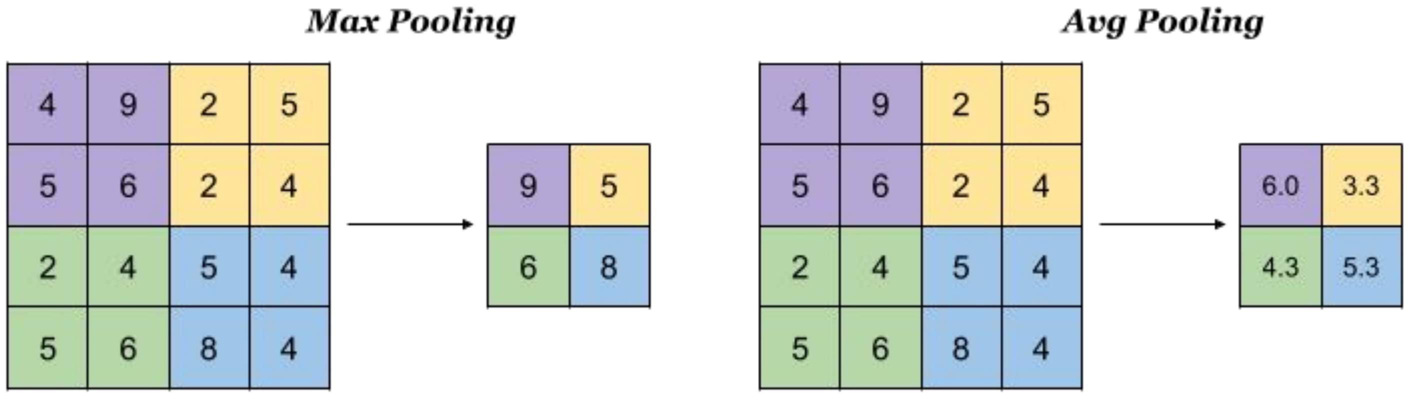


Convolution layer

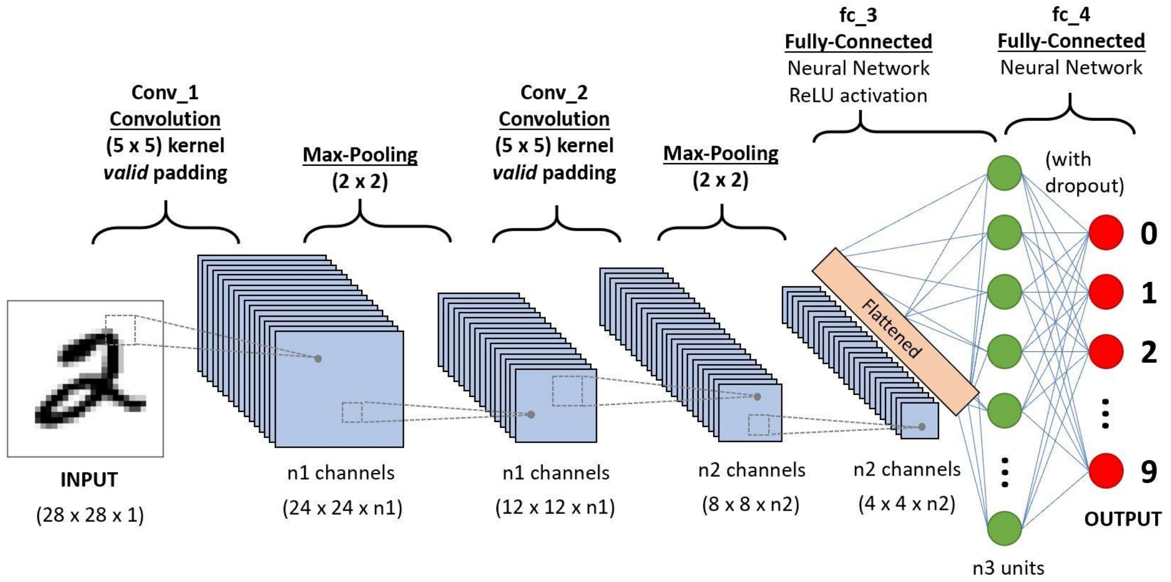
- Comments on the paper:
  - For short sentence classification, best region sizes are generally found between 1 and 10, and in practice, nf filters (with nf ∈ [100, 600]) are applied to each region to give the model the ability to learn different, complementary features for each region.
  - Since each filter generates a feature map, each region is thus embedded into an nf -dimensional space.
  - Moreover, using regions of varying size around the optimal one improves performance.
  - In that case, different parallel branches are created (one for each region size), and the outputs are concatenated after pooling.
  - Performance and cost increase with nf up to a certain point, after which the model starts overfitting

Pooling layer

- Reduces the dimensionality of the object being passed forward



Intercalating convolutional and pooling layers



Softmax layer

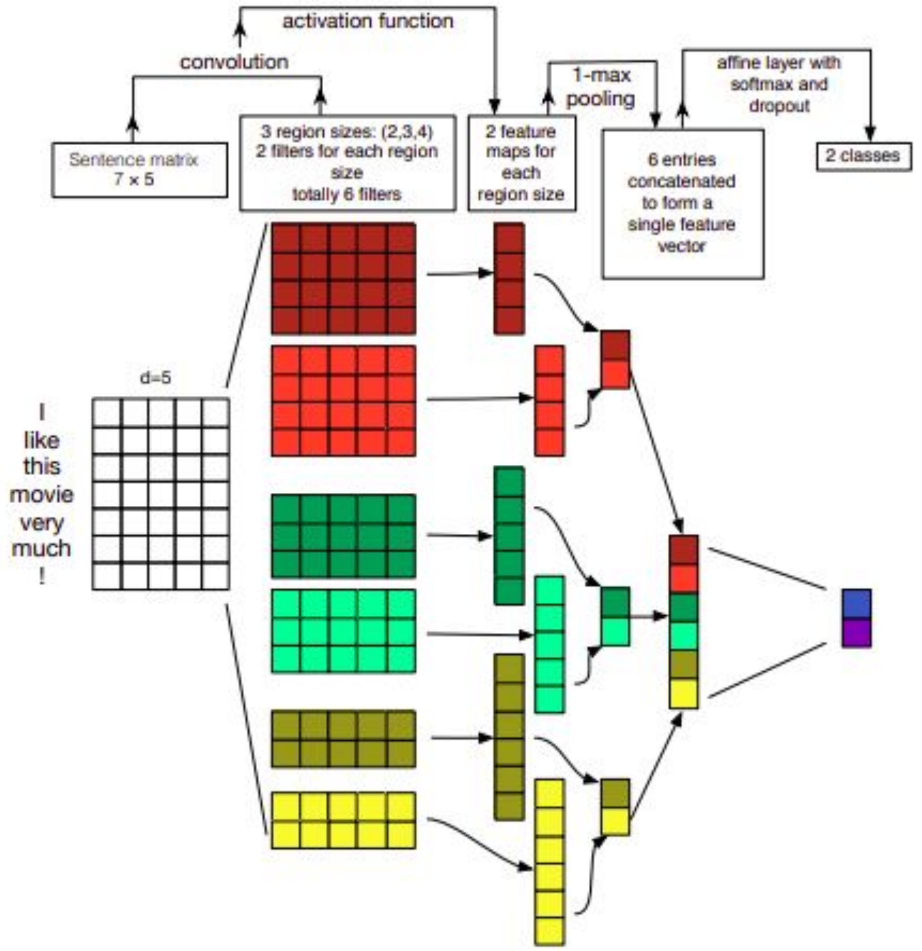
- A ***softmax function*** is applied to the document encoding to output class probabilities.

Softmax function

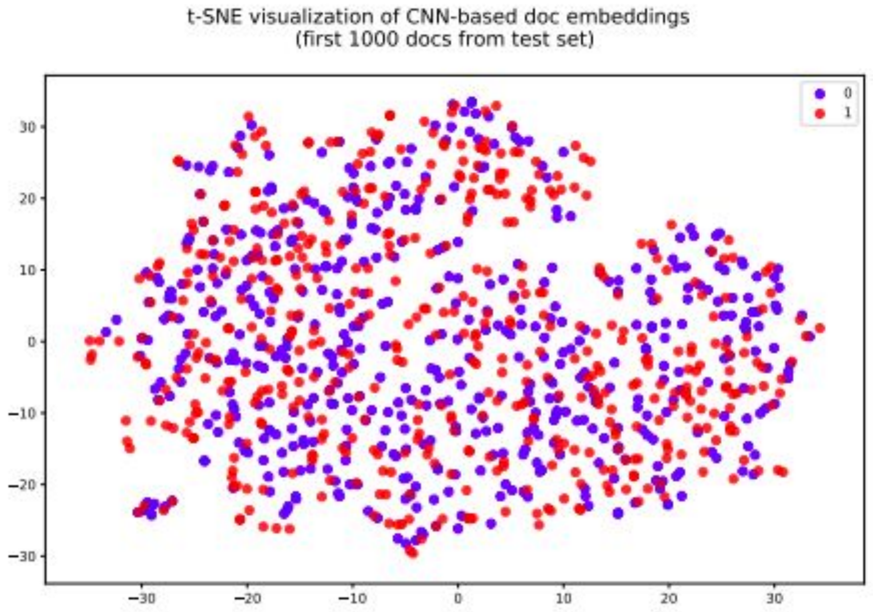
$$P(y=j \mid \Theta^{(i)}) = \frac{e^{\Theta^{(i)}}}{\sum_{j=0}^{k} e^{\Theta_k^{(i)}}}$$

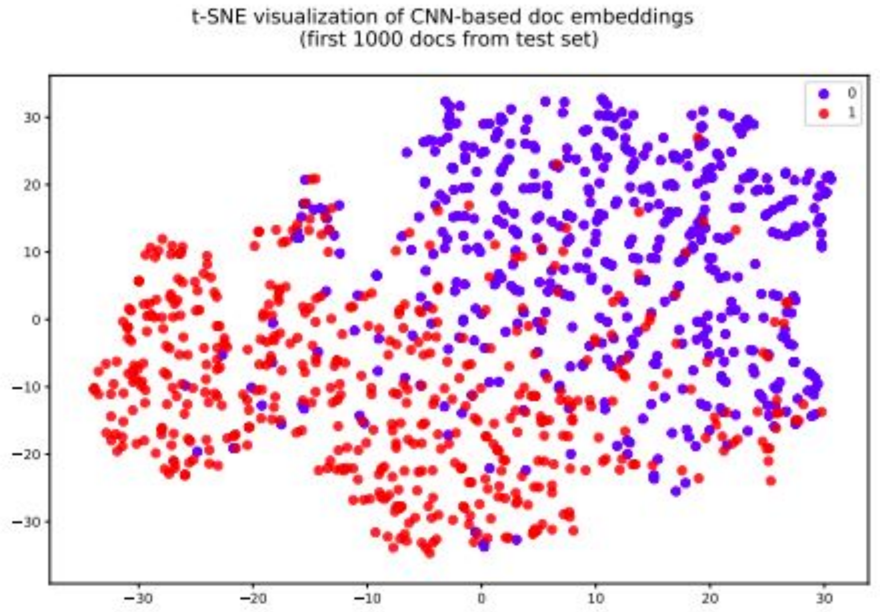
where $\Theta = w_0x_0 + w_1x_1 + \ldots + w_kx_k = \sum_{i=0}^{k} w_i x_i = w^T x$

CNN architecture for (short) document classification

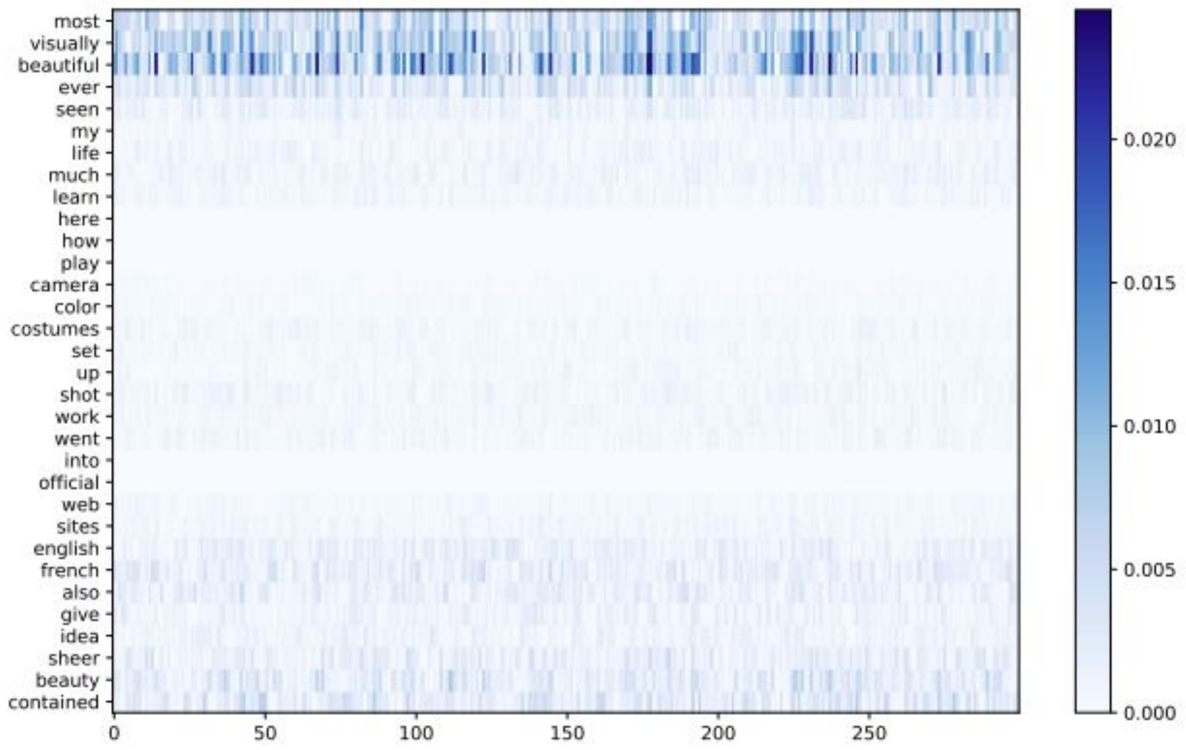

activation function
convolution
1-max pooling
affine layer with softmax and dropout
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 entries concatenated to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

Doc embeddings before training and after 2 epochs.

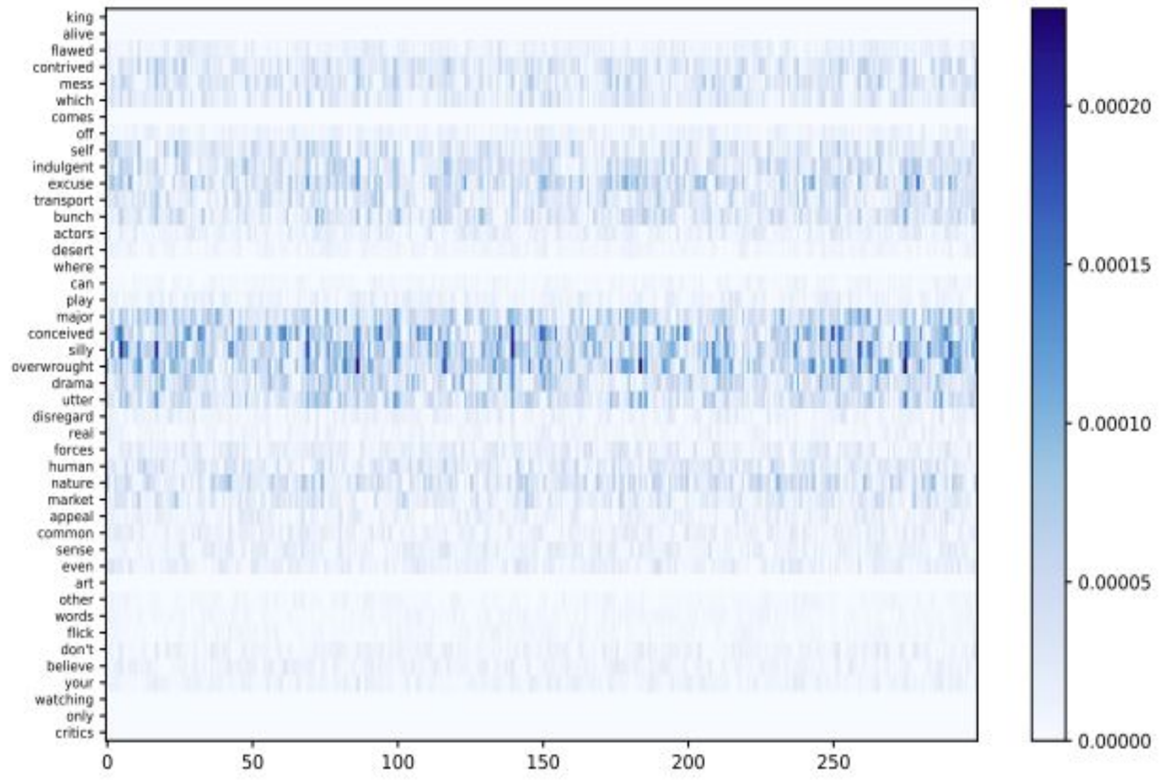




Saliency Maps - true label positive



Saliency Maps - true label negative

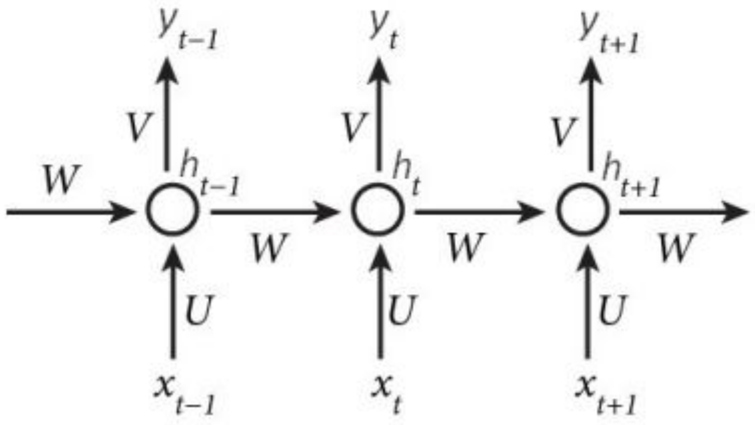


Recurrent Neural Networks (RNNs)

RNN framework

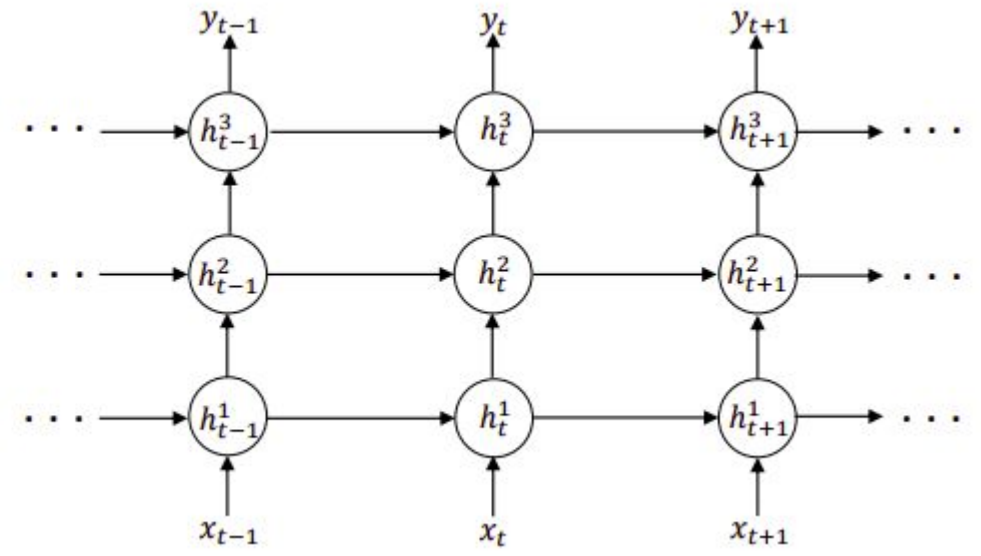
- RNNs were specifically developed to be used with sequences
- Some examples include time series, or, in NLP, words (sequences of characters) or sentences (sequences of words).
- CNNs do allow to capture some order information, but it is limited to local patterns, and long-range dependencies are ignored
- RNN can be viewed as a chain of simple neural layers that share the same parameters

3 steps of an unrolled RNN



A RNN can be viewed as a chain of simple neural layers that share the same parameters.

3 steps of an unrolled deep RNN



Each circle represents a RNN unit. The hidden state of each unit in the inner layers (1 & 2) serves as input to the corresponding unit in the layer above.

RNN framework

- At any step t in the sequence, the hidden state ht is defined in terms of the previous hidden state ht−1 and the current input vector xt in the following recursive way:

$$h_t = f(Ux_t + Wh_{t-1} + b)$$

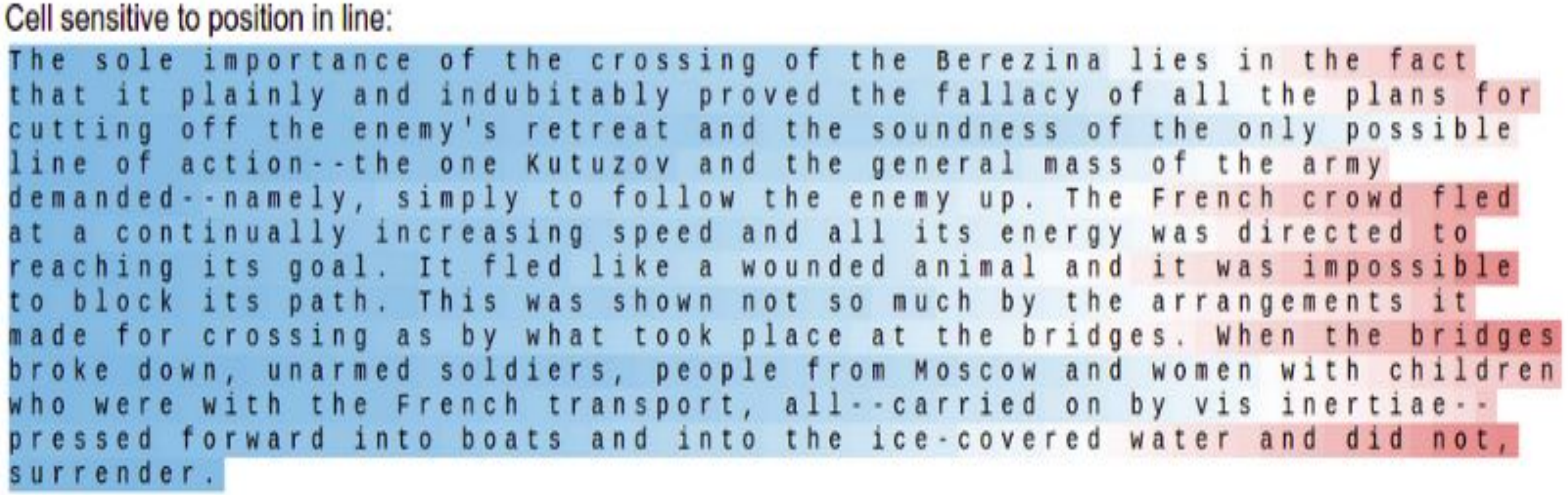
- The larger hidden layer, the greater the capacity of the memory, with an increase in computational cost.
- The output vector y transforms the current hidden state h in a way that depends on the final task. For classification, it is computed as:

$$y_t = \mathrm{softmax}(Vh_t)$$

Language modeling

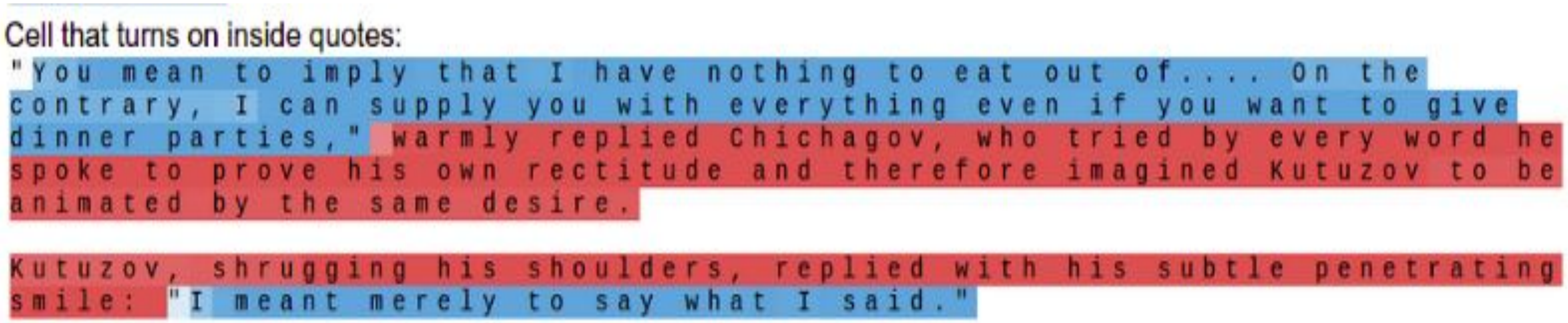
- Language modeling is a special case of classification where the model is trained to predict the next word or character in the sentence.
- At each time step t, the output vector gives the probability distribution of x over all the words/characters in the vocabulary, conditioned on the previous words/characters in the sequence.
- The language model can also be used to generate text of arbitrary size by repeatedly sampling characters for the desired number of time steps (for character-level granularity) or until the special end-of-sentence token is selected.

Language modeling


Cell sensitive to position in line:

```
The sole importance of the crossing of the Berezina lies in the fact
that it plainly and indubitably proved the fallacy of all the plans for
cutting off the enemy's retreat and the soundness of the only possible
line of action--the one Kutuzov and the general mass of the army
demanded--namely, simply to follow the enemy up. The French crowd fled
at a continually increasing speed and all its energy was directed to
reaching its goal. It fled like a wounded animal and it was impossible
to block its path. This was shown not so much by the arrangements it
made for crossing as by what took place at the bridges. When the bridges
broke down, unarmed soldiers, people from Moscow and women with children
who were with the French transport, all--carried on by vis inertiae--
pressed forward into boats and into the ice-covered water and did not,
surrender.
```

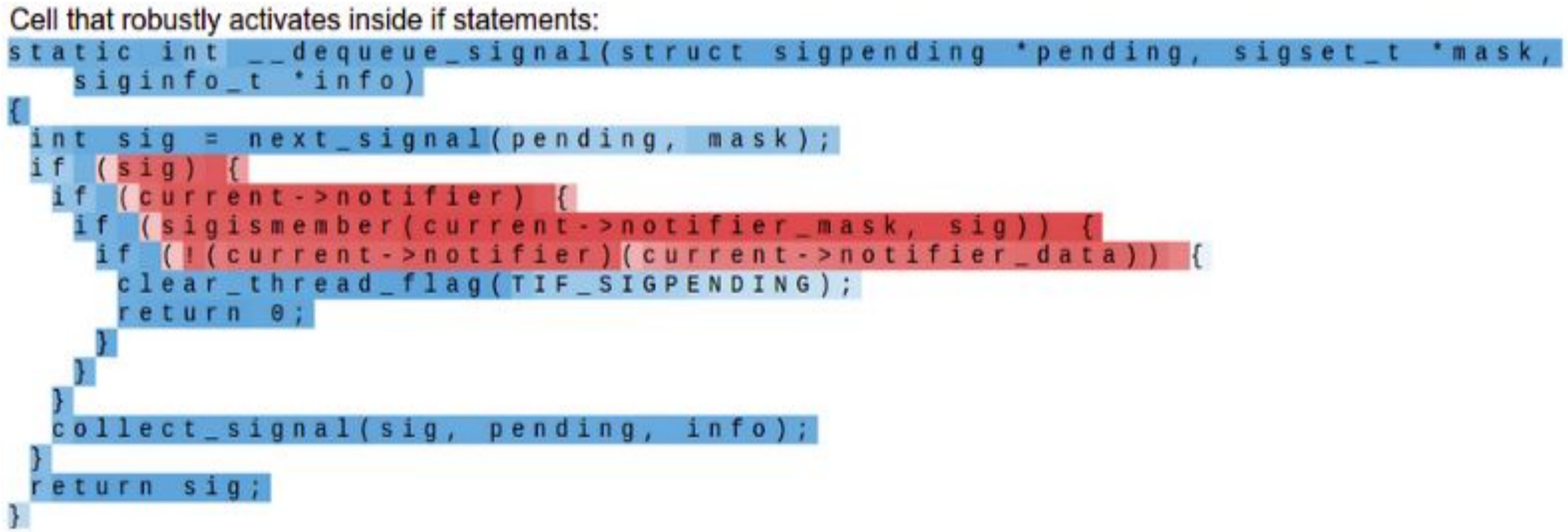
Language modeling


Cell that turns on inside quotes:

"You mean to imply that I have nothing to eat out of.... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

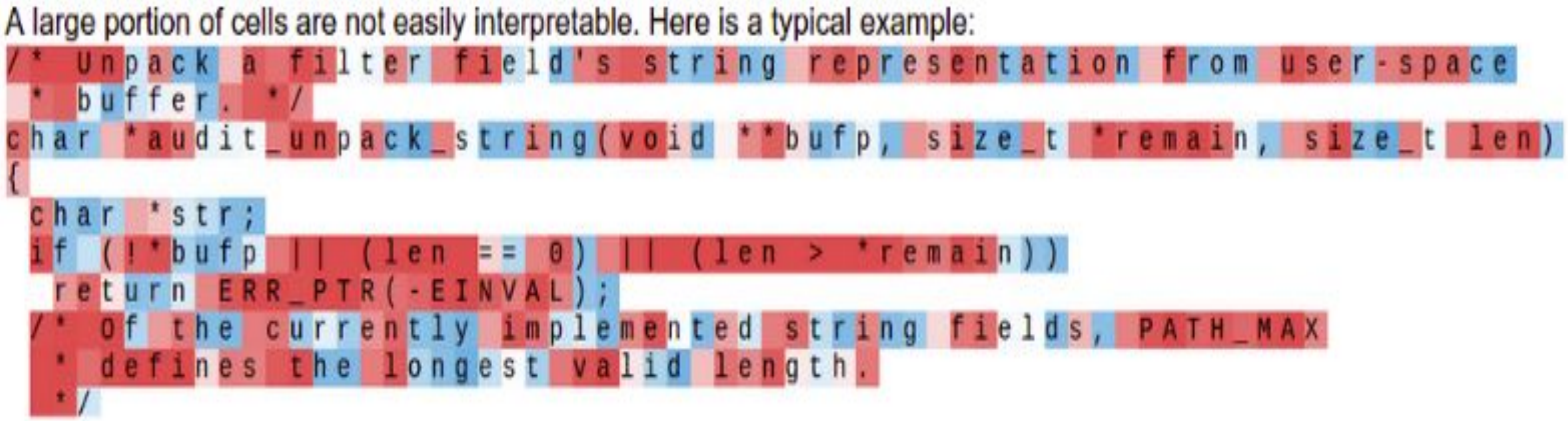
Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

Language modeling


Cell that robustly activates inside if statements:

```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
    siginfo_t *info)
{
 int sig = next_signal(pending, mask);
 if (sig) {
  if (current->notifier) {
   if (sigismember(current->notifier_mask, sig)) {
    if (!(current->notifier)(current->notifier_data)) {
     clear_thread_flag(TIF_SIGPENDING);
     return 0;
    }
   }
  }
  collect_signal(sig, pending, info);
 }
 return sig;
}
```

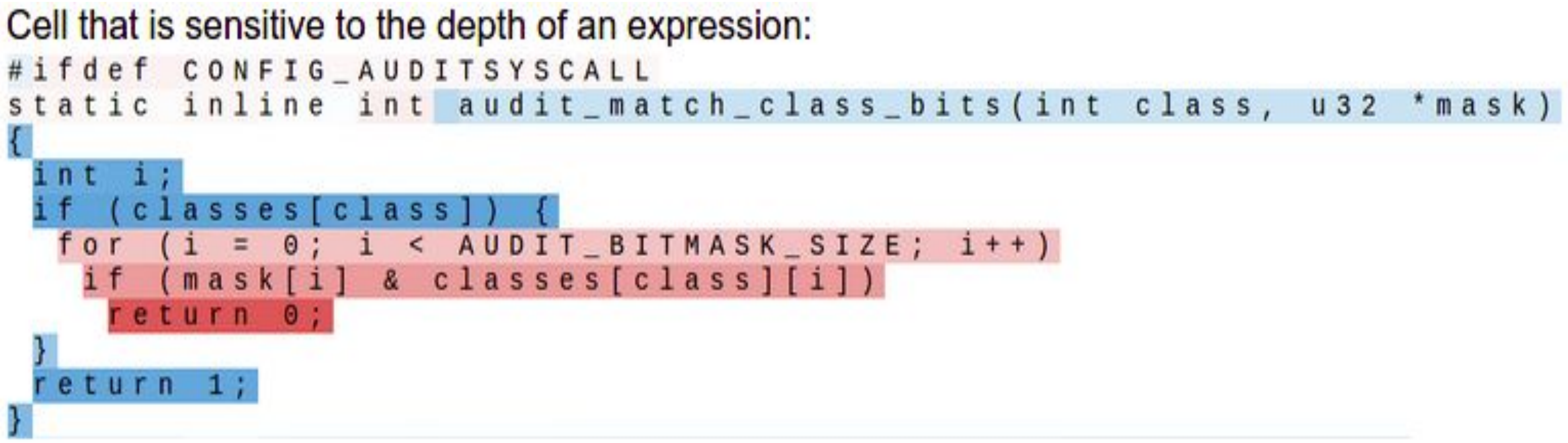
Language modeling



A large portion of cells are not easily interpretable. Here is a typical example:

```
/* Unpack a filter field's string representation from user-space
 * buffer. */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
  char *str;
  if (!*bufp || (len == 0) || (len > *remain))
    return ERR_PTR(-EINVAL);
  /* Of the currently implemented string fields, PATH_MAX
   * defines the longest valid length.
   */
```

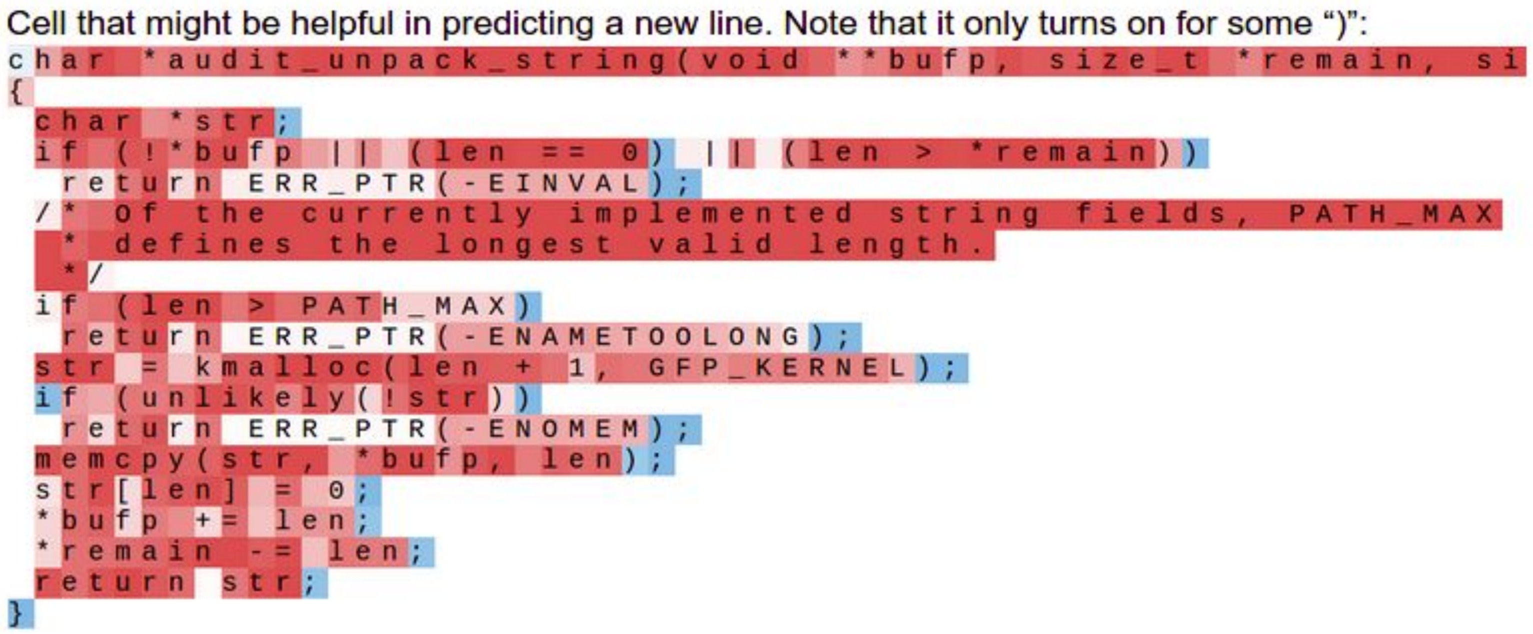
Language modeling



Cell that is sensitive to the depth of an expression:

```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
 int i;
 if (classes[class]) {
  for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
   if (mask[i] & classes[class][i])
    return 0;
 }
 return 1;
}
```

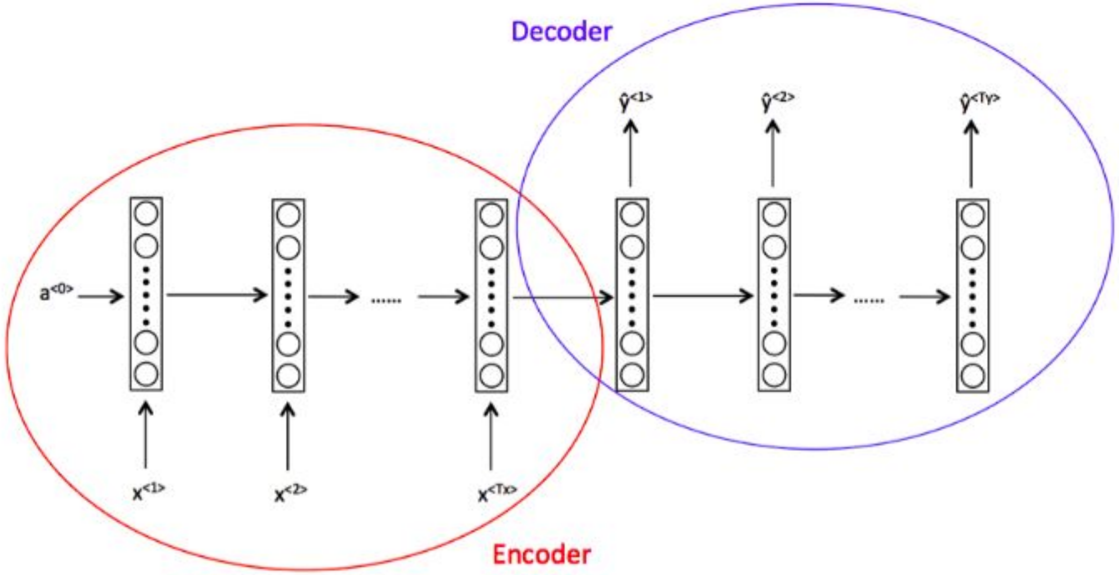
Language modeling



Cell that might be helpful in predicting a new line. Note that it only turns on for some “)”:

```
char *audit_unpack_string(void **bufp, size_t *remain, si
{
  char *str;
  if (!*bufp || (len == 0) || (len > *remain))
    return ERR_PTR(-EINVAL);
  /* Of the currently implemented string fields, PATH_MAX
   * defines the longest valid length.
   */
  if (len > PATH_MAX)
    return ERR_PTR(-ENAMETOOLONG);
  str = kmalloc(len + 1, GFP_KERNEL);
  if (unlikely(!str))
    return ERR_PTR(-ENOMEM);
  memcpy(str, *bufp, len);
  str[len] = 0;
  *bufp += len;
  *remain -= len;
  return str;
}
```

Language modeling



Picture Credit- Google

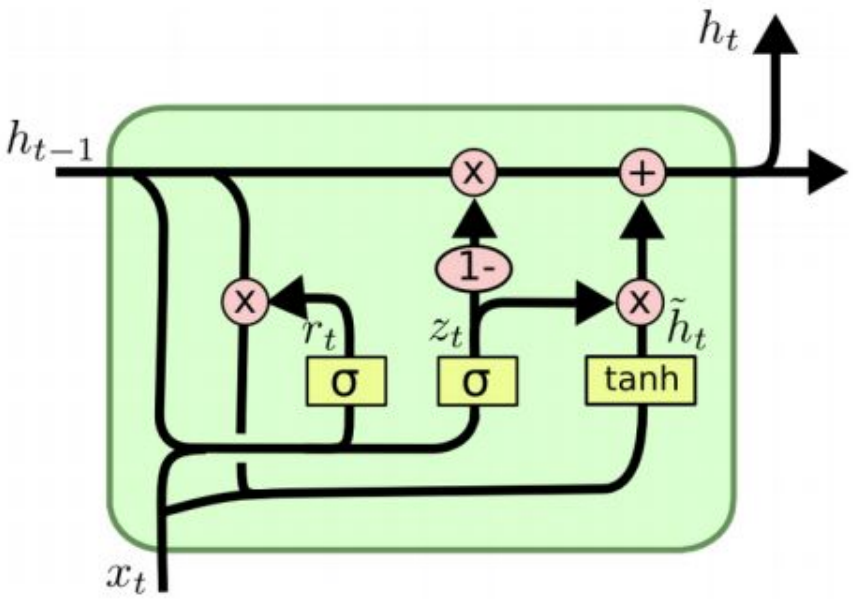
<u>Encoder</u> refers to the part of the network which reads the sentence to be translated, and,

<u>Decoder</u> is the part of the network which translates the sentence into desired language.

Limitations of RNN

- Apart from all of its usefulness RNN does have certain limitations major of which are :
    - Examples of RNN architecture stated above are capable of capturing the dependencies in only one direction of language. Basically in case of Natural Language Processing it assumes that the word coming after has no effect on the meaning of the word coming before. With our experience of languages we know that it is certainly not true.
    - RNN are also not very good in capturing long term dependencies and the problem of vanishing gradients resurface in RNN.
- Both these limitations give rise to new types of RNN architectures

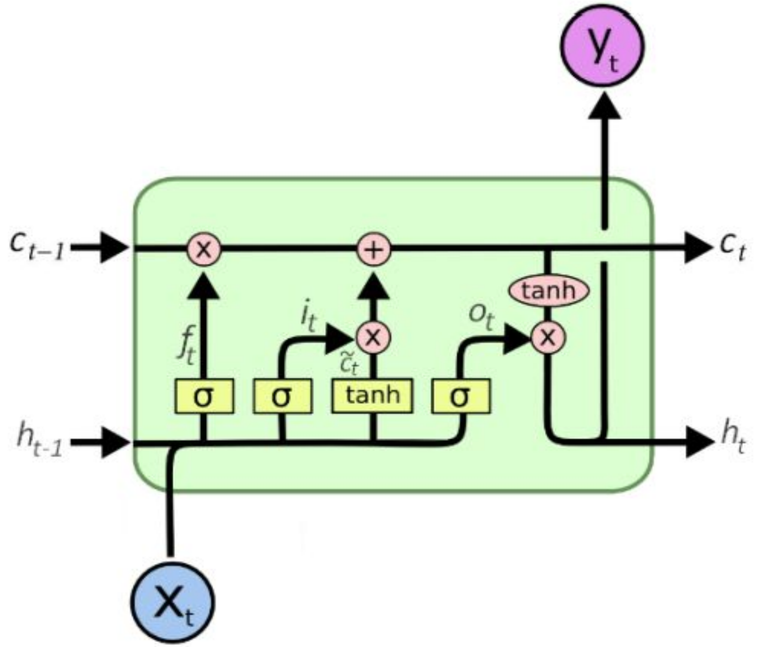
GRU unit



The GRU unit is a simplified LSTM unit with only two gates (reset and update), and where there is no explicit memory ct.

It helps to capture long range dependencies and also help a lot in fixing vanishing gradient problem.

The LSTM unit



With the LSTM unit, the hidden state is computed by four interacting layers that give the network the ability to remember or forget specific information about the preceding elements in the sequence.

In LSTM architecture instead of having one update gate as in GRU there is an update gate and a forget gate.

IMDB Movie review dataset

Overview

- The task is to perform binary classification (positive/negative) on reviews from the Internet Movie Database (IMDB) dataset.
- Known as ***sentiment analysis*** or ***opinion mining***.
- The dataset contains 50.000 movies reviews, labeled by polarity.
- 50% for ***Training*** and 50% for ***Testing***.
- Each review is a list of word indexes (integers) from a dictionary of size V where the most frequent word has index 1.

Binary classification objective function

The log loss (cross entropy) is defined as:

$$logloss = \frac{-1}{N}\sum_{i=1}^{N}(y_i log p_i + (1-y_i) log(1-p_i))$$

where N is the number of observations, p is the probability assigned to class 1, (1-p) is the probability assigned to class 0 and y is the true label of the i-th observation (0 or 1).

Now let’s do some coding…

(see examples on notebook)

https://github.com/NovaisGabriel/CNN_RNN_for_NLP

Thank you!