MADMO

RNN for texts

Тарас Хахулин
Deep Learning Engineer, Samsung AI Center
Skoltech, MIPT
Tg: @vitaminotar
https://github.com/khakhulin/

Let us drive right in!

image taken from [1]

What is the NLP?

NLP is a subfield of Artificial Intelligence (AI) which relies on Computer Science and Linguistic

The goal is to make computers understand and generate natural language to perform useful tasks like:

– **Translate a text** from one language to another, e.g. Yandex Translate

– **Search and extract information**

  Search engines, e.g. Google

  Question answering systems, e.g. IBM Watson

– **Dialogue systems**

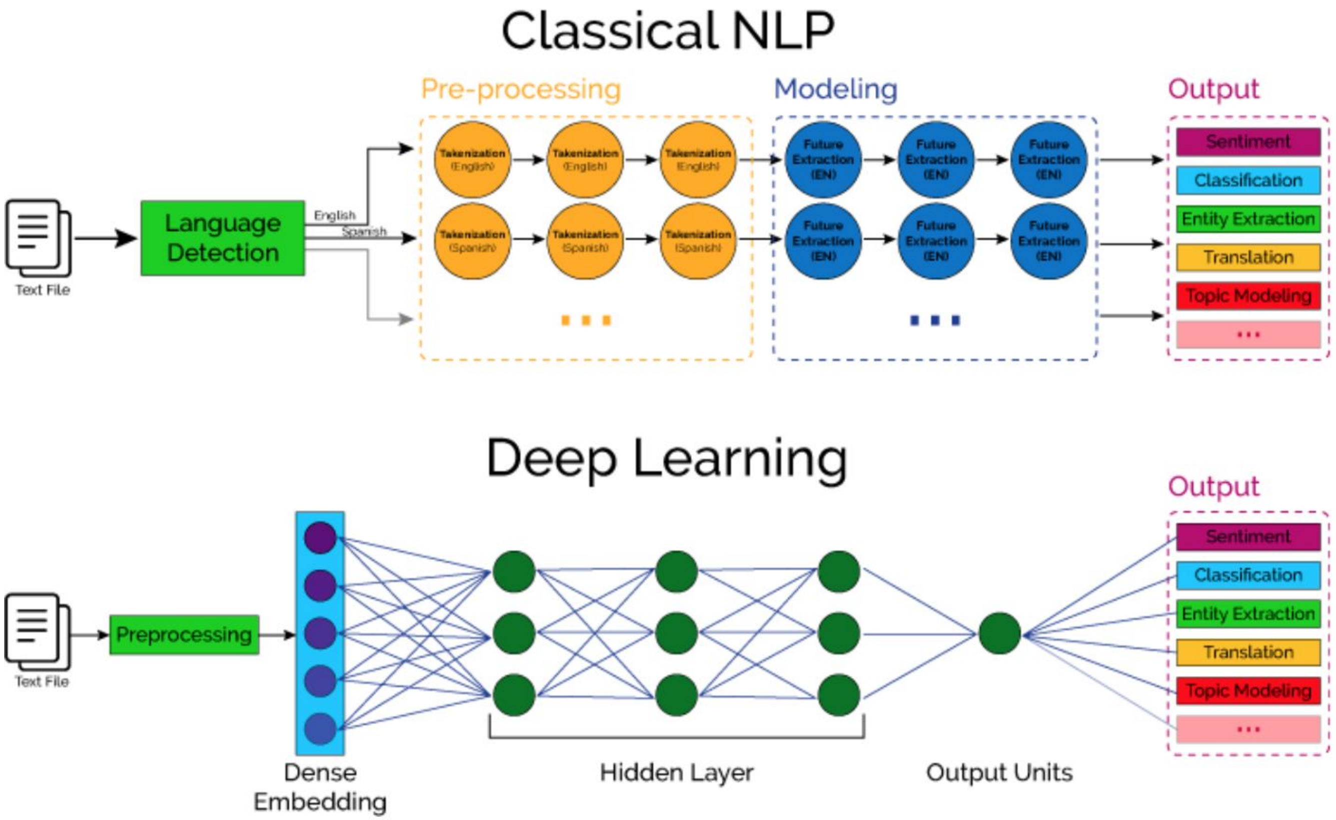
  Answer questions, execute voice commands, voice typing

  Samsung Bixby, Apple Siri, Google Assistant, etc.

– **Language understanding** is an “AI-complete” problem

  we hope to train computers to extract signal relevant for a particular task

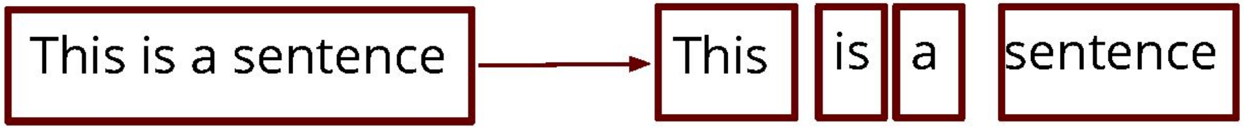
Remineder about pipelines



https://www.upwork.com/hiring/for-clients/artificial-intelligence-and-natural-language-processing-in-big-data/

Preprocessing

- Tokenization: split the input into tokens



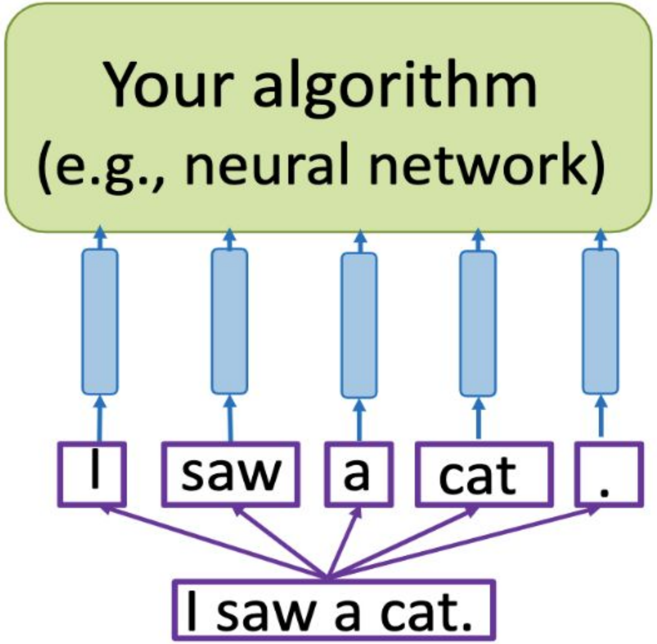
Preprocessing

- Token normalization

Dog, dogs → dog

Bark, barks → bark

Distributional semantics



Any algorithm for solving any task

Word representation - vector
(word embedding)

Sequence of tokens

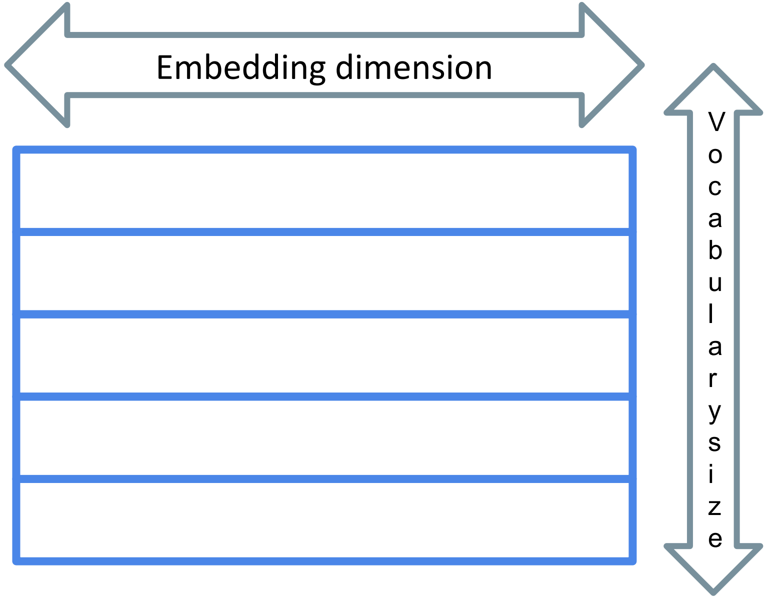
Text

Distributional semantics

I | saw | a | cat | . |

31| 1237 | 12 | 139 | 9 |

Token indices in the vocabulary

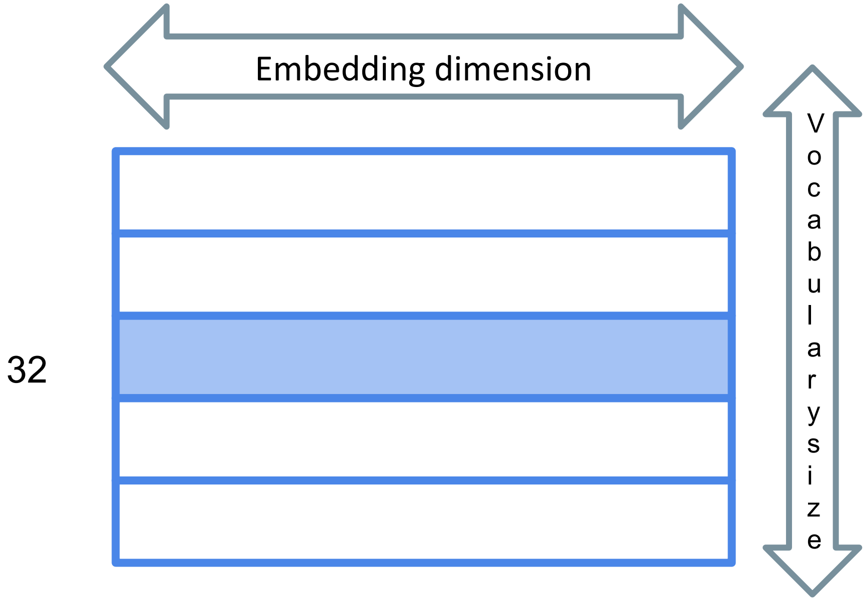


Simple Table

Distributional semantics

I | saw | a | cat | . |

32| 1237 | 12 | 139 | 9 |

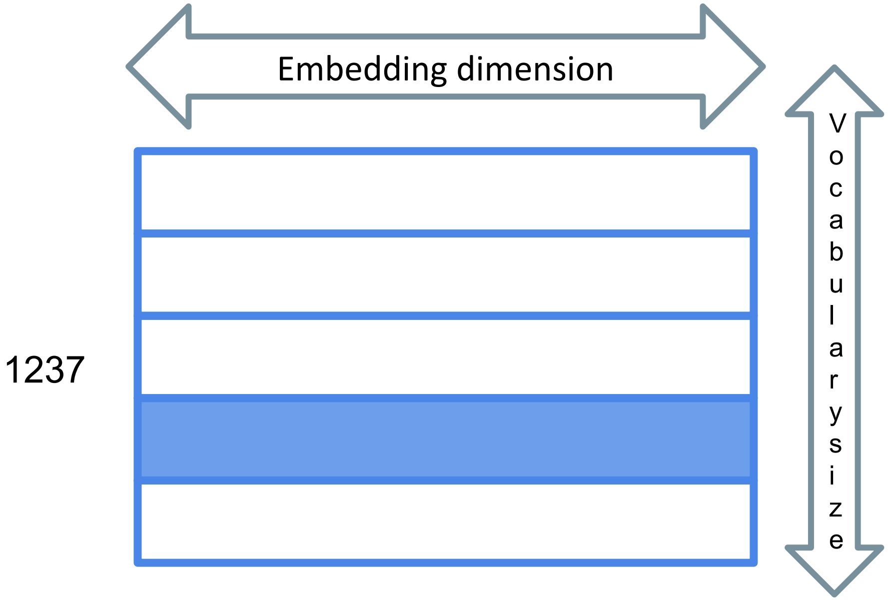


Simple Table

Distributional semantics

I | saw | a | cat | . |

32| 1237 | 12 | 139 | 9 |

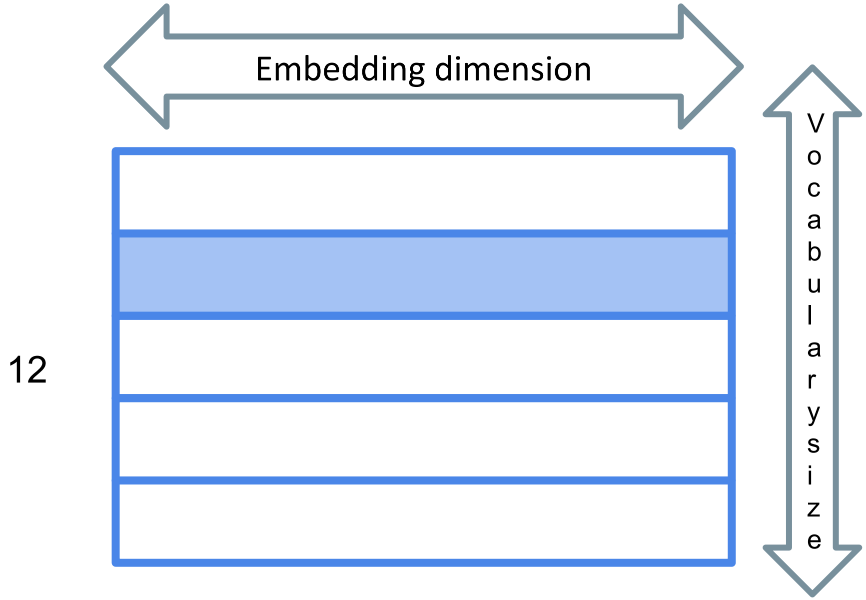


Simple Table

Distributional semantics

I | saw | a | cat | . |

32| 1237 | 12 | 139 | 9 |



Simple Table

What is inside?

What is inside: one-hot

We can encode every word as a one hot vector (500,000).

dog = [0...0,1,0...0]

cat = [1,0...0]

What is the problem?

What is inside: one-hot

We can encode every word as a one-hot vector (500,000).

dog = [0...0,1,0...0]

cat = [1,0...0]

What is the problem?

BUT

- There is no any semantic in such vectors
- There is no comparision type

Distributional semantics

Does vector similarity imply semantic similarity?

Distributional semantics

Does vector similarity imply semantic similarity?

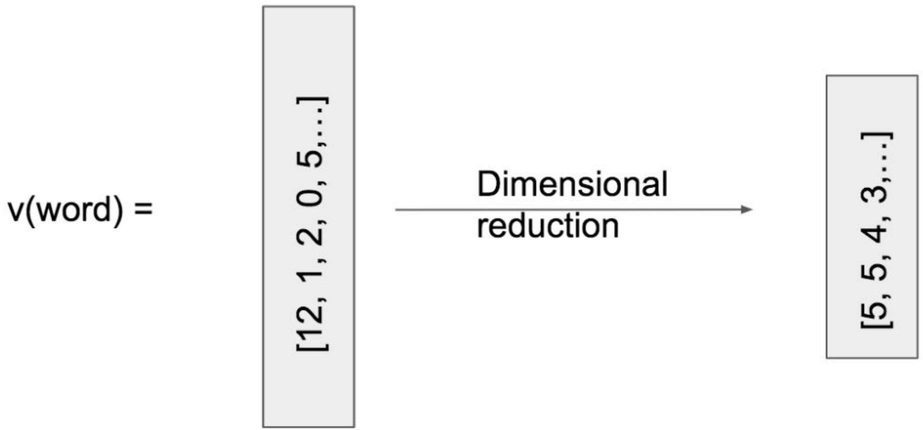
The distribution hypothesis, stated by Firth (1957):
"You shall know the word by the company it keeps"

Co-occurrence Counts

v(word_i)[j] = count(co-occurrences word_i with word_j)

v(word) = [12,1,2,0,5,...]

The same problem with the size



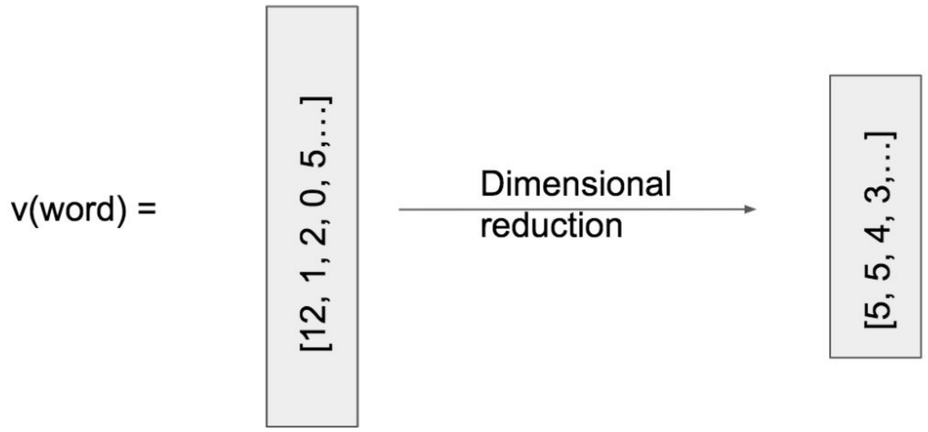
Co-occurrence Counts

v(word_i)[j] = count(co-occurrences word_i with word_j)

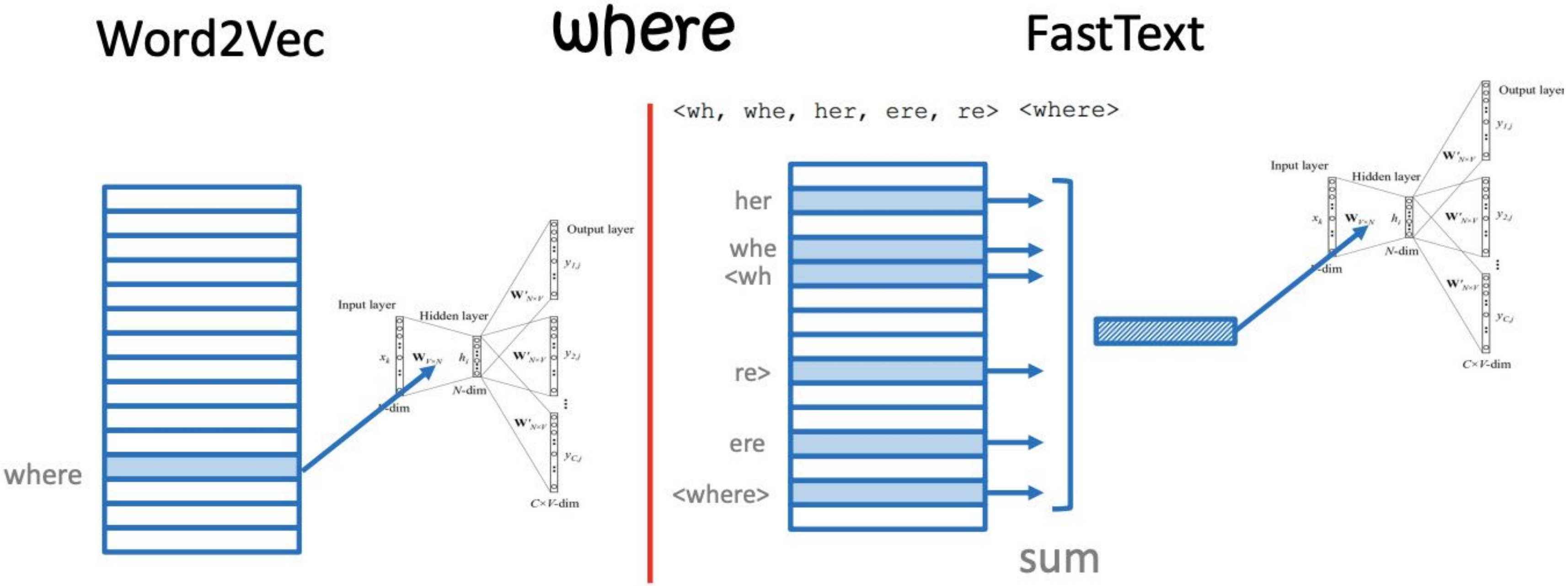
v(word) = [12,1,2,0,5,...]

The same problem with the size

We can compress with different technique on the matrices



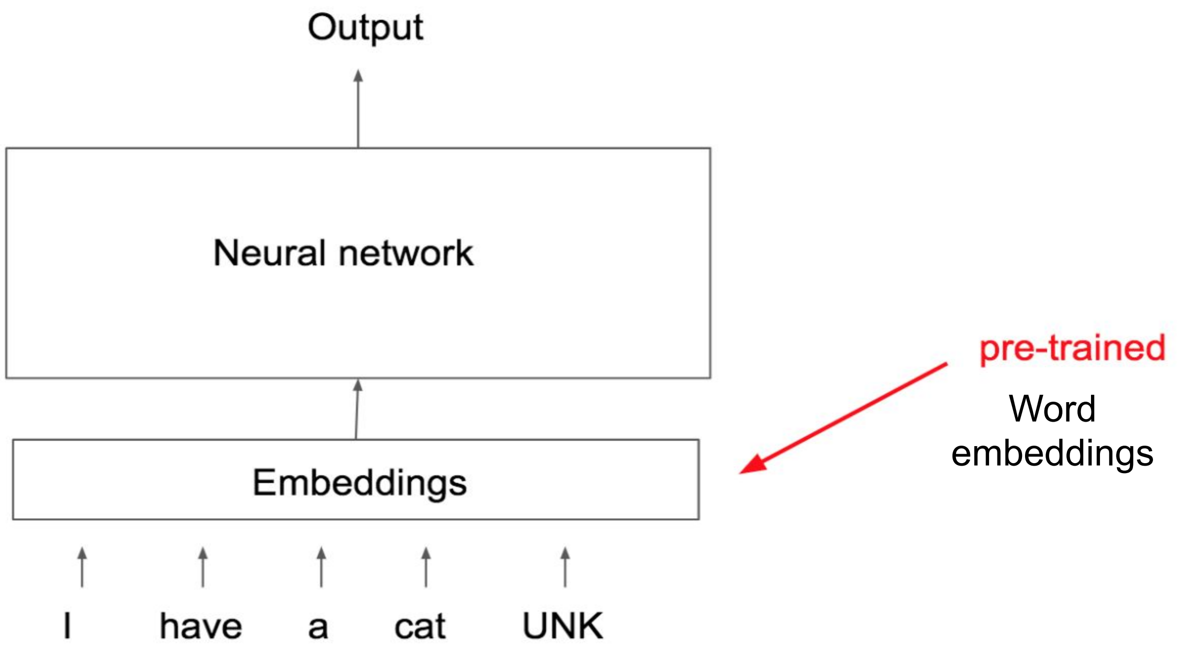
Word2Vec vs FastText



Source: [6]

Why do we need such vectors?

When you have small text data for your task



Where could we see LMs?

Basic Intuition

"Probability of a sentence" = how likely is it to occur in natural language

$$\text{P(the cat slept peacefully)} > \text{P(slept the peacefully cat)}$$

$$\text{P(she studies morphosyntax)} > \text{P(she studies more faux syntax)}$$

Language model

Given sequence of words LM can predict probability of this sequence.

I have a very interesting idea. -> P(I,have,a,very,interesting,idea)

- It is difficult to know the true probability of an arbitrary sequence of words (at least they are changing in time)
- We need to approximate probability with some model
- As usual this model will be better in one bunch of things and worse in another

- We can do language model statistically
- We can use ML algorithms

N-gram Language Models

We want to estimate $P(s = w_1 \dots w_n)$.
Example: $P(s = the\ cat\ slept\ quietly)$

N-gram Language Models

We want to estimate $P(s = w_1 \ldots w_n)$.

Example: $P(s = the\ cat\ slept\ quietly)$

This is really a joint probability over the words in s:

$$P(w_1 = the, w_2 = cat, w_3 = slept, w_4 = quietly)$$

N-gram Language Models

We want to estimate $P(s = w_1 \dots w_n)$.
Example: $P(s = the\ cat\ slept\ quietly)$

$$P(X, Y) = P(X|Y)P(Y)$$

This is really a joint probability over the words in s:

$P(w_1 = the, w_2 = cat, w_3 = slept, w_4 = quietly)$

$$P(the\ cat\ slept\ quietly) = P(quietly|the\ cat\ slept) \cdot P(the\ cat\ slept) =$$
$$P(quietly|the\ cat\ slept) \cdot P(slept \mid the\ cat) \cdot P(cat|the) \cdot P(the)$$

N-gram Language Models

The chain rule gives us:

$$P(w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n} P(w_i | w_1 \ldots w_{i-1})$$

Problems?

N-gram Language Models

The chain rule gives us:

$$P(w_1, w_2, \dots, w_n) = \prod_{i=1}^{n} P(w_i | w_1 \dots w_{i-1})$$

Problems?

Many of these conditional probs are just as sparse!

If we want $P(I\ spent\ three\ years\ before\ the\ mast\ \dots)$

we still need $P(mast \mid I\ spent\ three\ yers\ before\ the)$

N-gram Language Models: solution

We make an independence assumption:

the probability of a word only depends on a fixed number of previous words (history).

Markov property:

$$P(w_i|w_1, w_2, \dots, w_{i-1}) = P(w_i|w_{i-n+1}, \dots, w_{i-1})$$

Examples:

– trigram model: $P(w_i|w_0, w_1, \dots, w_{i-1}) \approx P(w_i|w_{i-2}, w_{i-1})$
– bigram model: $P(w_i|w_0, w_1, \dots, w_{i-1}) \approx P(w_i|w_{i-1})$
– unigram model: $P(w_i|w_0, w_1, \dots, w_{i-1}) \approx P(w_i)$

N-gram Language Models: general

$$P(w_i|w_1,\dots,w_{i-1}) = P(w_i|\underbrace{w_{i-n+1},\dots,w_{i-1}}_{\text{(n-1) words}}) =$$

prob of an n-gram

prob of an (n-1)-gram

$$= \frac{\boxed{P(w_i, w_{i-1},\dots,w_{i-n+1})}}{\boxed{P(w_{i-1},\dots,w_{i-n+1})}}$$

Question: How do we get these n-gram and (n-1)-gram probabilities?

N-gram Language Models: general

$$P(w_i|w_1,\dots,w_{i-1}) = P(w_i|\overbrace{w_{i-n+1},\dots,w_{i-1}}^{\text{(n-1) words}}) =$$

prob of an n-gram

prob of an (n-1)-gram

$$= \frac{\boxed{P(w_i, w_{i-1},\dots,w_{i-n+1})}}{\boxed{P(w_{i-1},\dots,w_{i-n+1})}}$$

Answer: By counting them in some large corpus of text!

$$\approx \frac{count(w_i, w_{i-1},\dots,w_{i-n+1})}{count(w_{i-1},\dots,w_{i-n+1})}$$

Statistical approximation

How to evaluate Language Models?

Cross-entropy and Perplexity

For $(w_1 w_2 \dots w_n)$ with large *n*, per-word cross-entropy is well approximated by:

$$H_M(w_1 w_2 \dots w_n) = -\frac{1}{n} \cdot \log P_M(w_1 w_2 \dots w_n)$$

Lower cross-entropy ⇒ model is better at predicting next word.

Perplexity (reported in papers):

$$\text{perplexity} = 2^{cross-entropy}$$

How we can create a Language model using Neural Networks

Fixed size solution:

Sample probability of next token, given fixed size input:

$$label = softmax(Wx + b)$$

We can concatenate word embeddings:

$$label = softmax(W[\text{cat,seat,on}] + b)$$

We still have problems with fixed size model...

Neural Language Models?

Output distribution

$\hat{y} = \text{softmax}(Uh + b_2) \in \mathbb{R}^{|V|}$

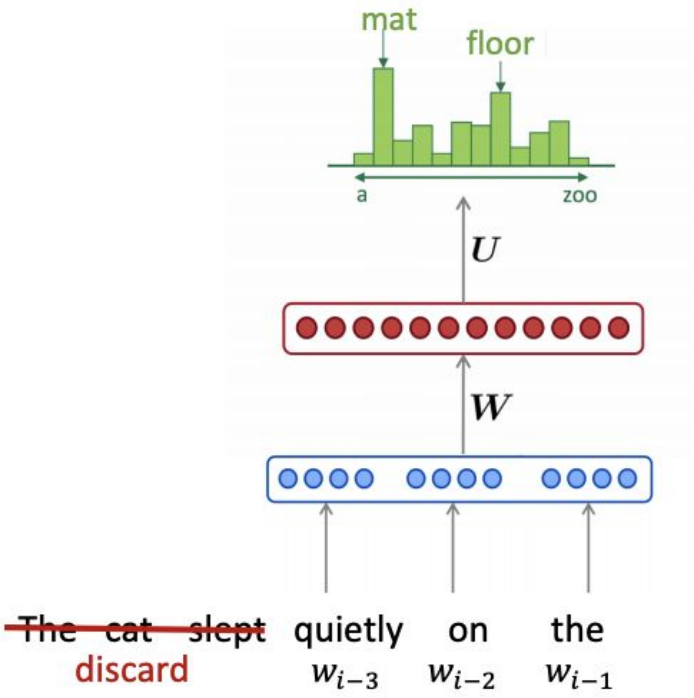
Hidden layer (or any feed-forward NN)

$h = f(Wx + b)$

Concatenate word embeddings

$x = (x_{i-3}, x_{i-2}, x_{i-1})$

Word embeddings



Neural Language Models?

Pros & Cons

Improvements over N-gram LMs:

- no sparsity problem
- model size O(n) (not O(exp(n)))

Remaining problems:

- Fixed window is too small
- Enlarging window enlarges W
- Window can never be large enough!
- Each uses different rows of . We don't share weights across the window.

Solution

We need a neural architecture that can process input of any length!

Neural Language Models?

Pros & Cons

Improvements over N-gram LMs:

- no sparsity problem
- model size O(n) (not O(exp(n)))

Remaining problems:

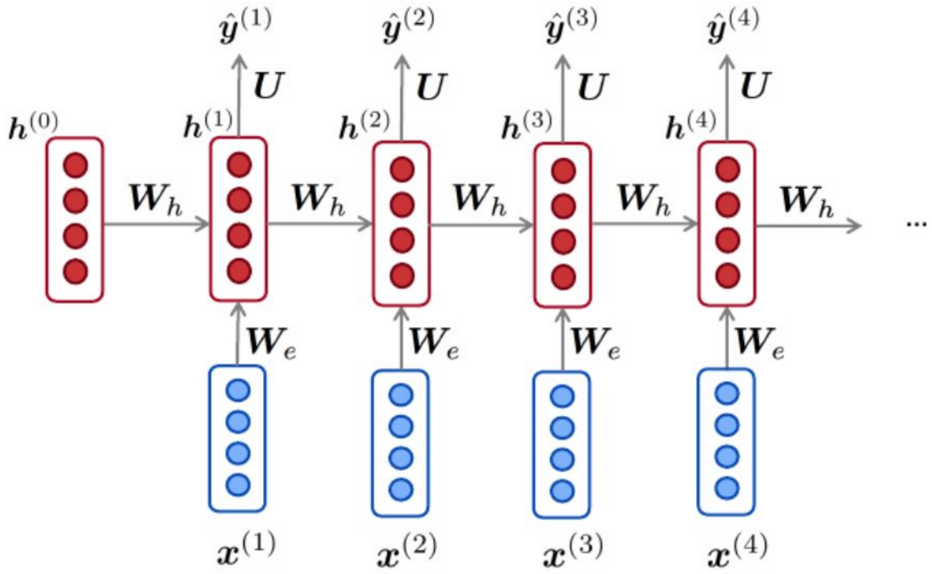
- Fixed window is too small
- Enlarging window enlarges W
- Window can never be large enough!
- Each uses different rows of . We don't share weights across the window.

Solution

We need a neural architecture that can process input of any length!

В общем случае мы хотим иметь доступ не только к n-граммам, но и ко всем предыдущим словам. Мы хотим каким-то образом запоминать информацию для предсказания текущего токена.

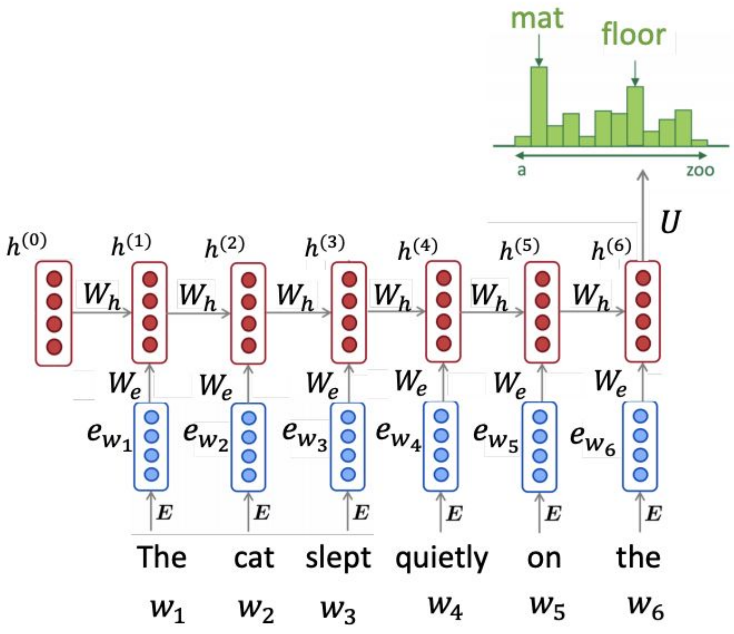
Recurrent neural networks (RNNs)



RNN story

RNN Advantages:

- Can process any length input
- Model size doesn't increase for longer input
- Computation for step t can (in theory) use information from many steps back
- representations shared across timesteps

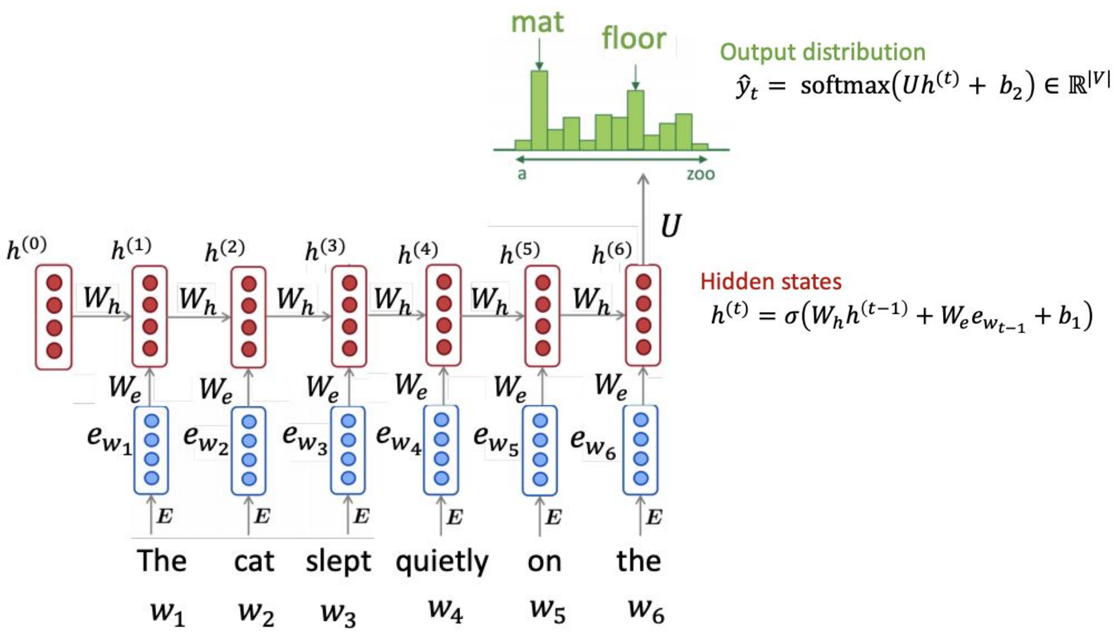


Source: [2]

RNN story

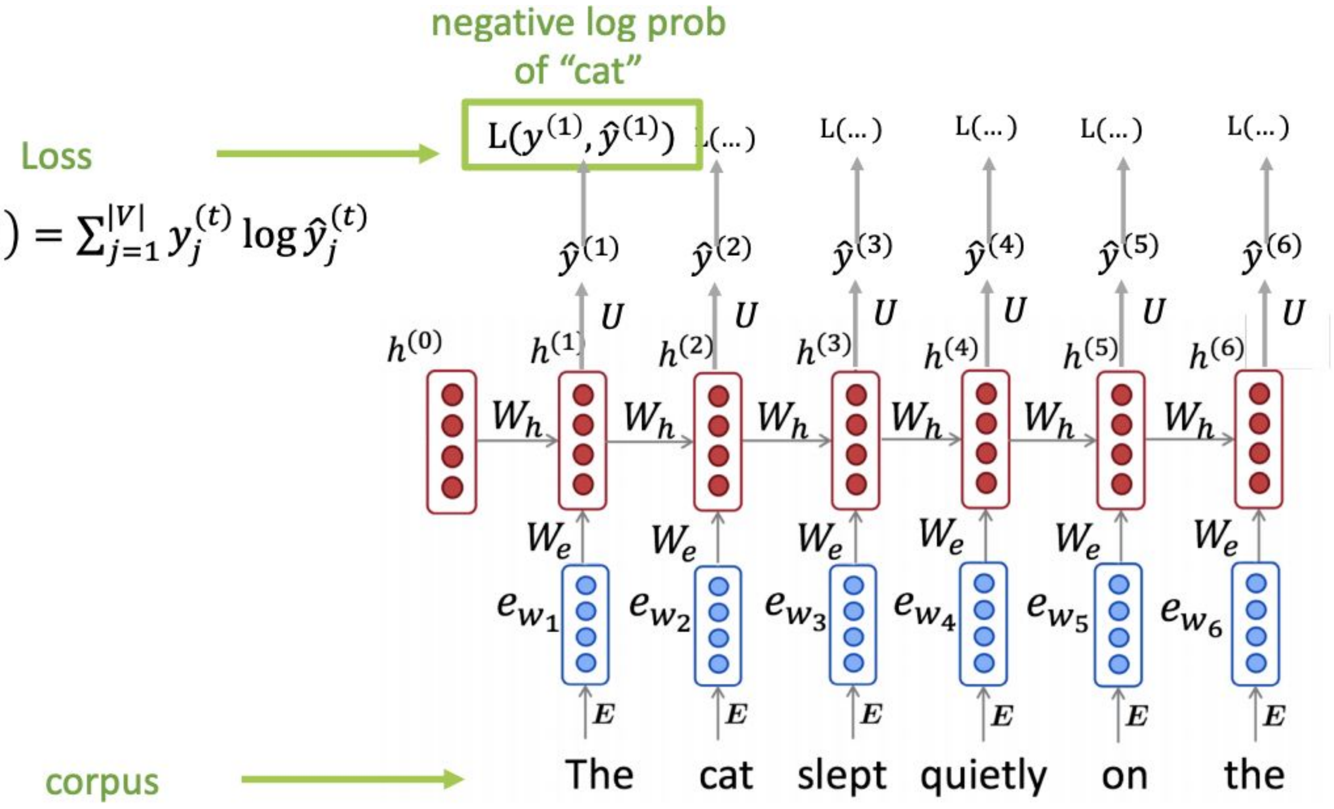
RNN **Advantages**:

- Can process any length input
- Model size doesn't increase for longer input
- Computation for step t can (in theory) use information from many steps back
- representations shared across timesteps



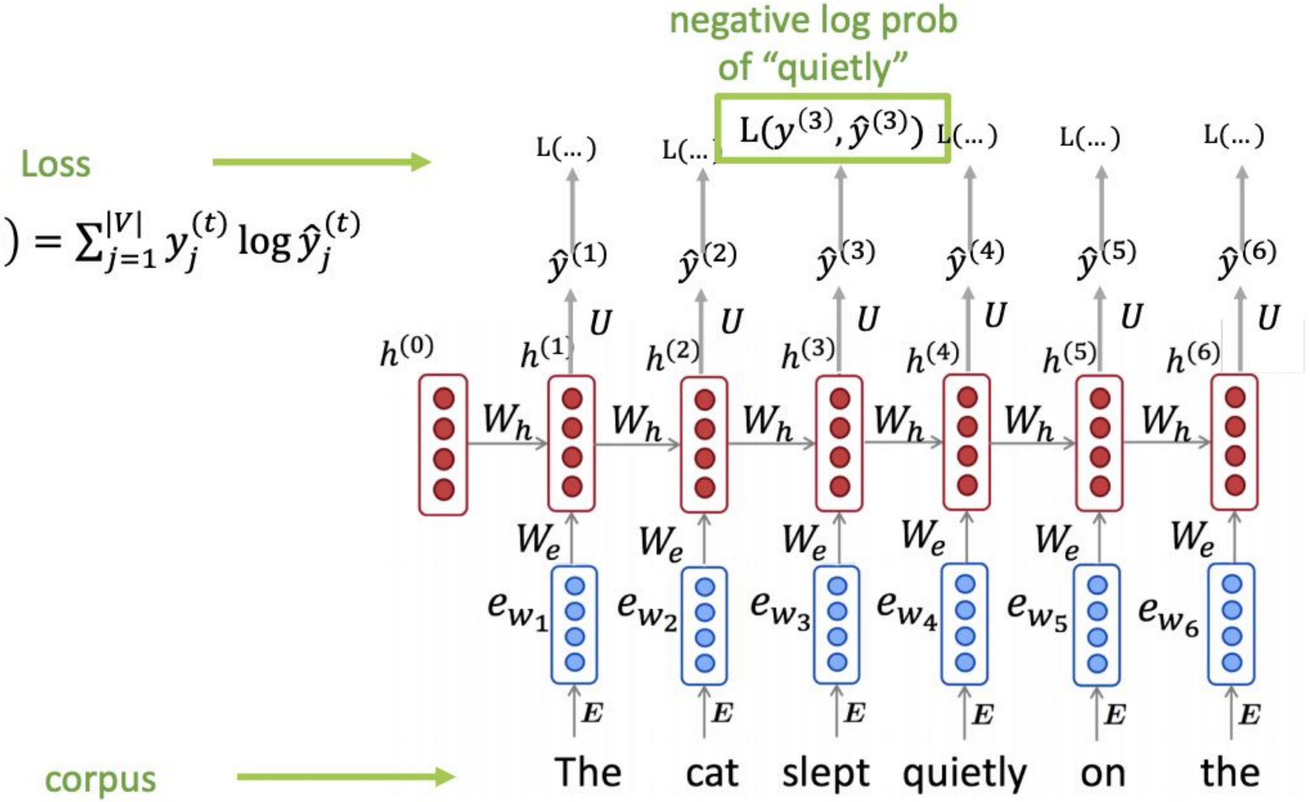
Language Models Training

$$L(y^{(t)}, \hat{y}^{(t)}) = \sum_{j=1}^{|V|} y_j^{(t)} \log \hat{y}_j^{(t)}$$



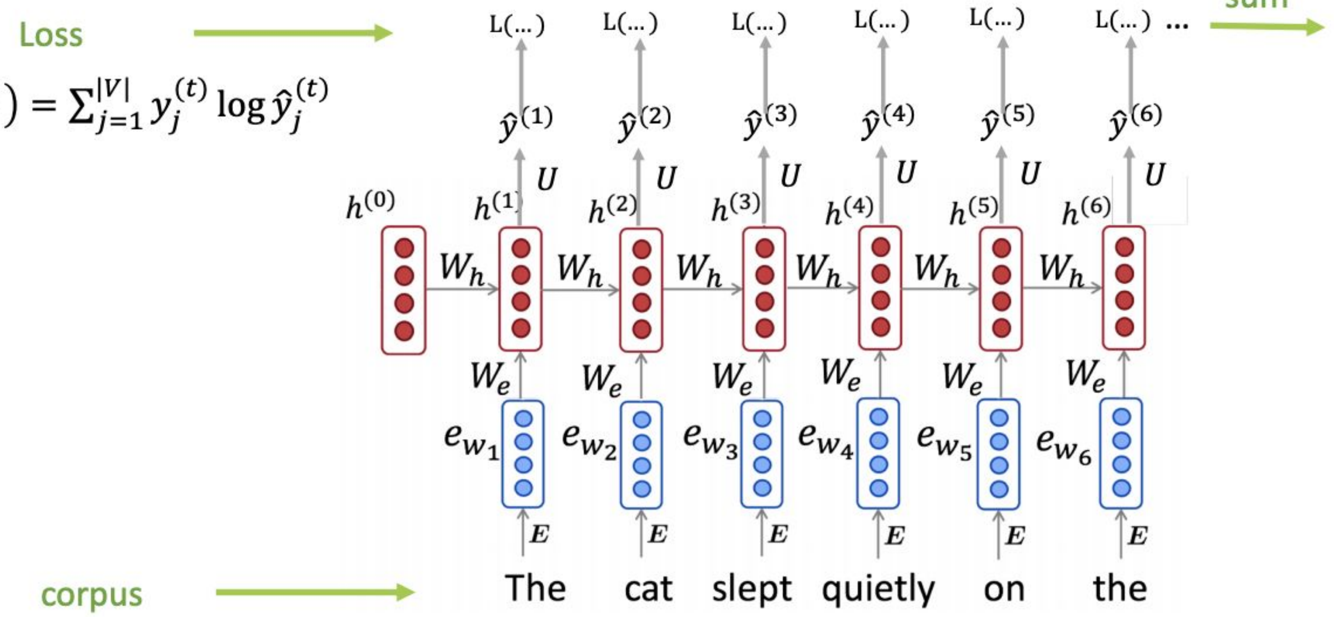
Language Models Training

$$L\left(y^{(t)}, \hat{y}^{(t)}\right) = \sum_{j=1}^{|V|} y_j^{(t)} \log \hat{y}_j^{(t)}$$



Language Models Training

$$L\left(y^{(t)}, \hat{y}^{(t)}\right) = \sum_{j=1}^{|V|} y_j^{(t)} \log \hat{y}_j^{(t)}$$



$$\sum_{t=1}^{T} L(y^{(t)}, \hat{y}^{(t)})$$

Small recap

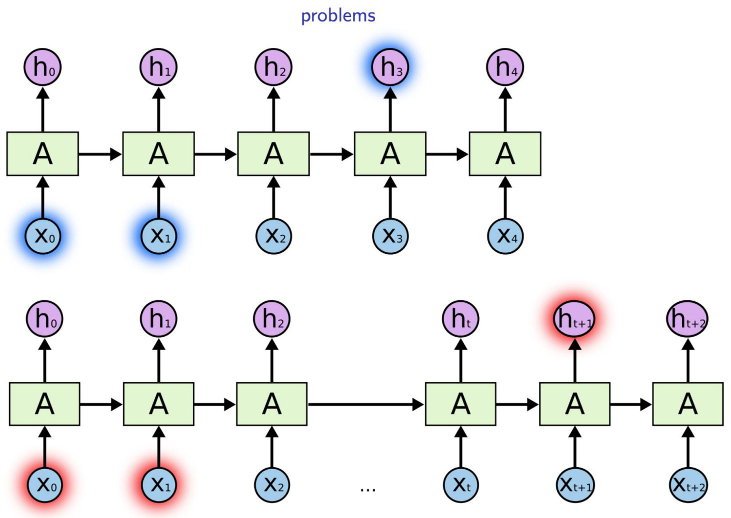
RNN Language model

Benefits:

- Can process any length input with the same model
- Can use information occured many steps before
- Share representations

Problems:

- Works quite slow
- In fact it is difficult to access remote information



New Era of RNNs

What do we have:

$$s_{t-1} = (s_1, s_2, \dots, s_n) - \textit{previous state}$$
$$h_{t-1} = (h_1, h_2, \dots, h_q) - \textit{previous output}$$
$$x_{t-1} = (x_1, x_2, \dots, x_p) - \textit{current input}$$

What do we want

What to forget:

What to remember:

What to output:

New Era of RNNs

What do we have:

$$s_{t-1} = (s_1, s_2, ..., s_n) - previous\ state$$
$$h_{t-1} = (h_1, h_2, ..., h_q) - previous\ output$$
$$x_{t-1} = (x_1, x_2, ..., x_p) - current\ input$$

What do we want:

What to forget: $\tilde{s}_{t-1} = f(x_t, h_{t-1}) \cdot s_{t-1}, F \in [0,1]^n$

What to remember: $\tilde{s}_t = r(x_t, h_{t-1}) \cdot m(x_t, h_{t-1})$

$r \in [0,1]^n, \ \ m \in [-1,1]^n$

new state: $s_t = \tilde{s}_{t-1} + \tilde{s}_t$

What to output: $h_t = act(W \cdot (x_t, h_{t-1}) + b) \cdot i(s_t)$

New Era of RNNs

What we do not have:

$f \in [0,1]^n$ – *forgetting function*

$r \in [0,1]^n, \; m \in [-1,1]^n$, *remembering function*

$i \in [-1,1]^n$ – *impact function*

What we can have:

$$f \in [0,1]^n F = \sigma(W_f \cdot (x_t, h_{t-1}) + b_f)$$

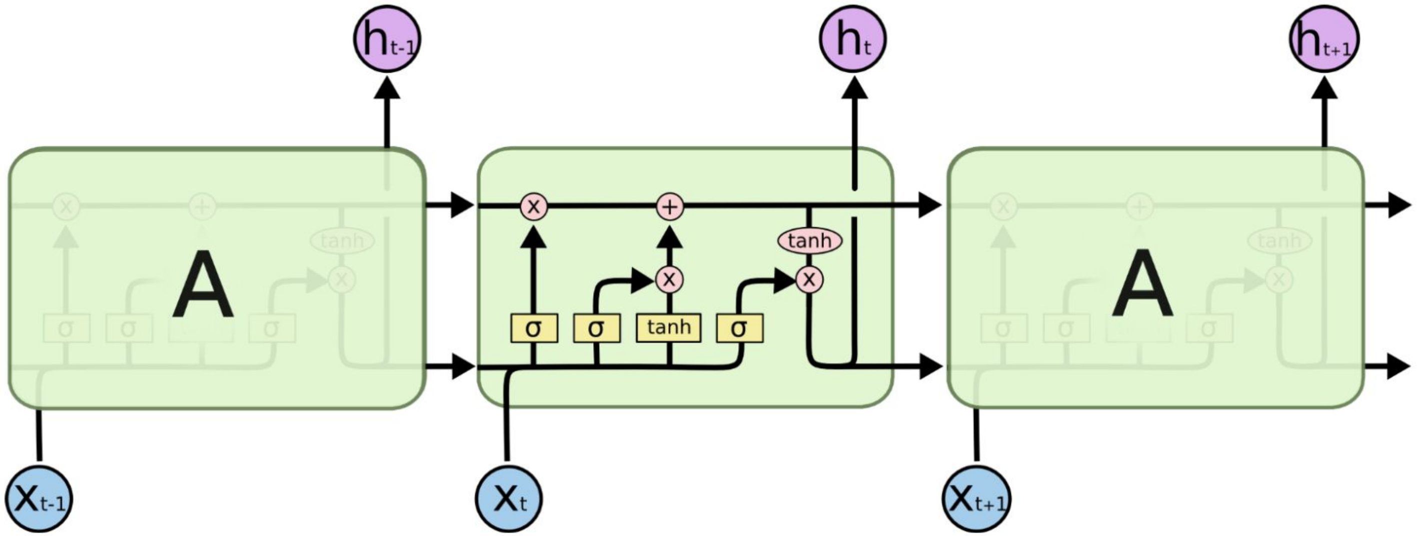
$$r \in [0,1]^n, \; m \in [-1,1]^n,$$

$$r = \sigma(W_r \cdot (x_t, h_{t-1}) + b_r),$$

$$m = tanh(W_m \cdot (x_t, h_{t-1}) + b_m)$$

$$i \in [-1,1]^n, \; i = tanh(s_t)$$

LSTM



Source: [3]

LM: FUN

Char-based LM trained on Linux Source Code

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    else
      ret = 1;
    goto bail;
  }
  segaddr = in_SB(in.addr);
  selector = seg / 16;
  setup_works = true;
  for (i = 0; i < blocks; i++) {
    seq = buf[i++];
    bpf = bd->bd.next + i * search;
    if (fd) {
      current = blocked;
    }
  }
  rw->name = "Getjbbregs";
  bprm_self_clearl(&iv->version);
  regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
  return segtable;
}
```

Source: [1]

LM: FUN

Char-based LM trained on Latex of book on algebraic geometry

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

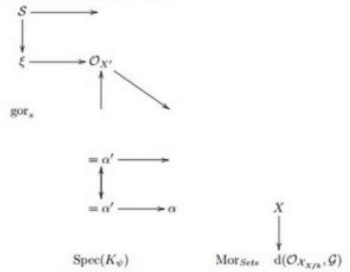
be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram



is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \quad \text{-1}(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_\ell}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{v}})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

More hallucinated algebraic geometry. Nice try on the diagram (right).

Source: [1]

LM: recap

Count-based (statistical)

- make an n-th order Markov assumption
- estimate n-gram probabilities (counting and smoothing)

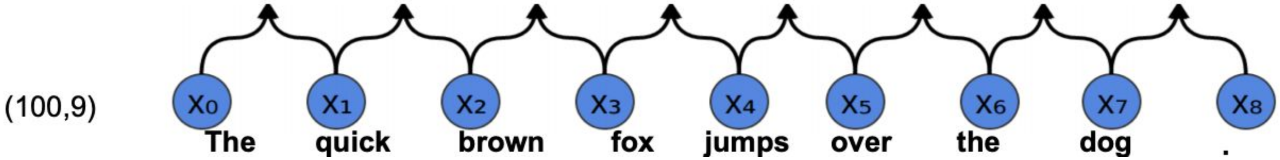
Neural Language Models

- solve the problem of data sparsity of the n-gram model, by representing words as vectors
- the parameters are learned as part of the training process

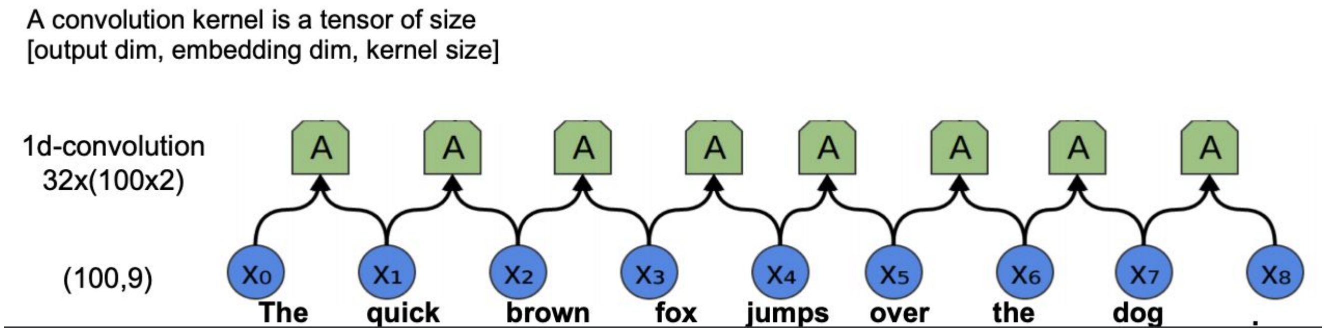
Source: [2]

More NLP neural networks?

CNN for texts

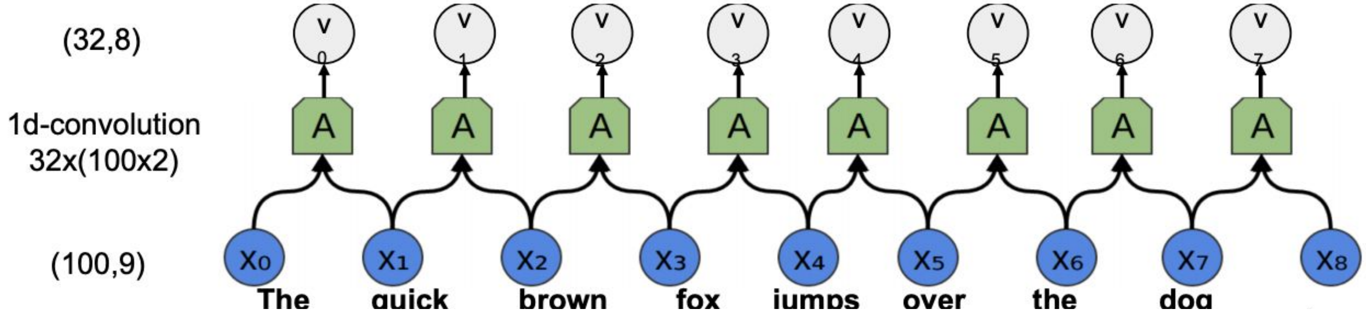


CNN for texts



CNN for texts

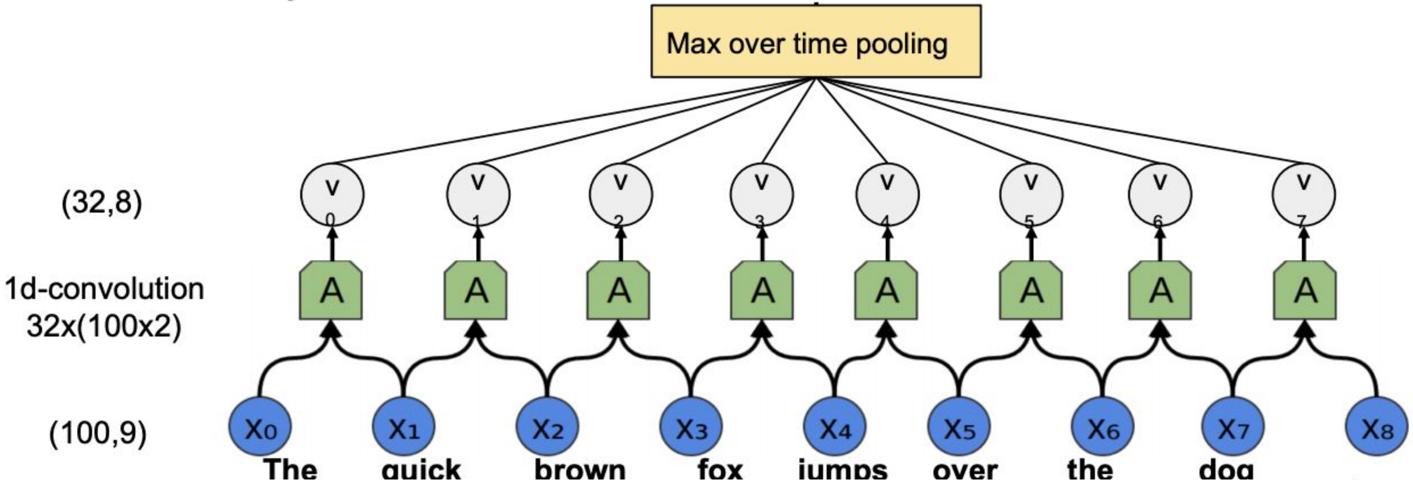
$$\mathbf{v}_0 = \mathbf{A}(\mathbf{x}_0, \mathbf{x}_1)$$
$$v_{0i} = A_i(\mathbf{x}_0, \mathbf{x}_1) = \sum_j (K_{0ij} x_{0j} + K_{1ij} x_{1j})$$



CNN for texts

$$\mathbf{v}_0 = \mathbf{A}(\mathbf{x}_0, \mathbf{x}_1)$$

$$v_{0i} = A_i(\mathbf{x}_0, \mathbf{x}_1) = \sum_j (K_{0ij}x_{0j} + K_{1ij}x_{1j})$$

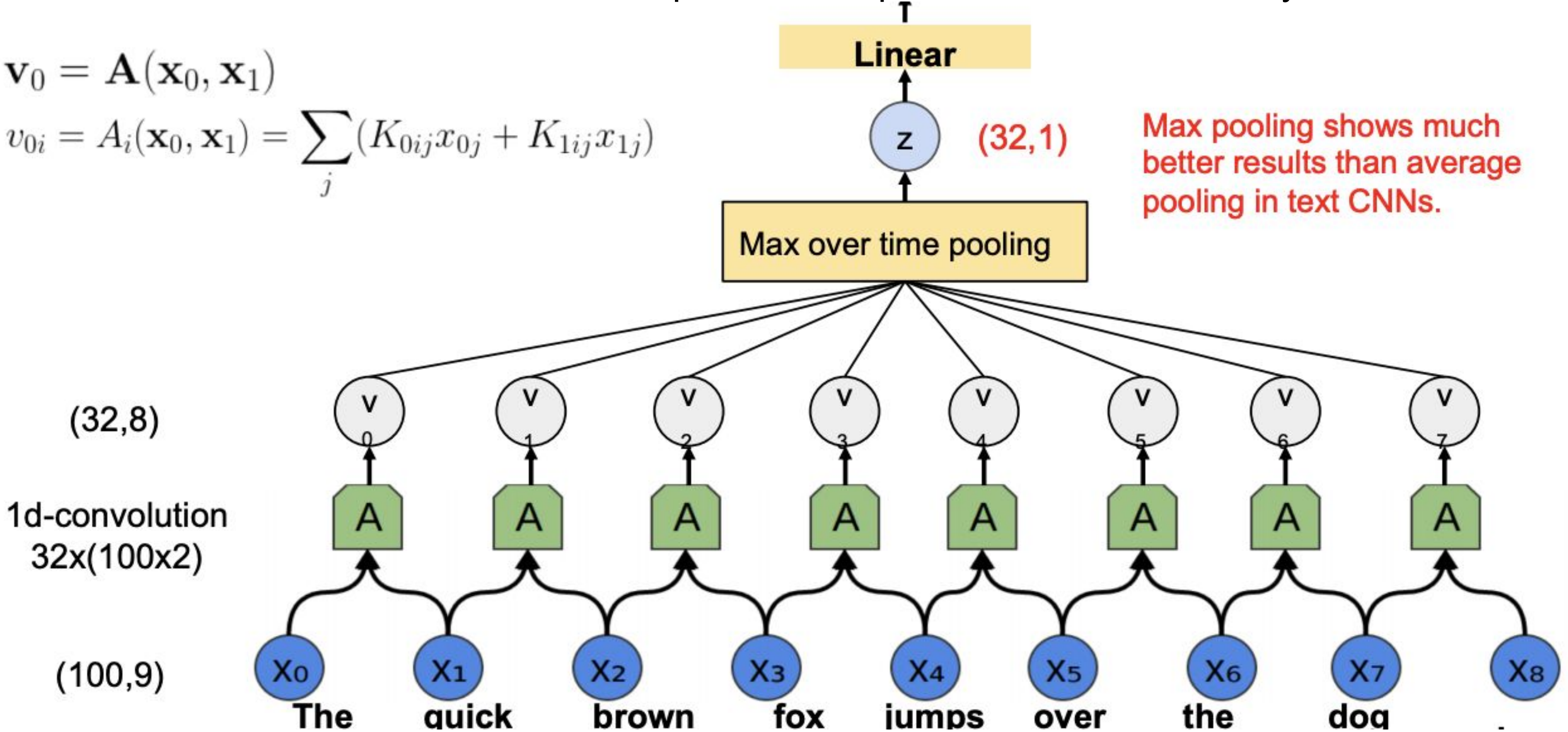


CNN for texts

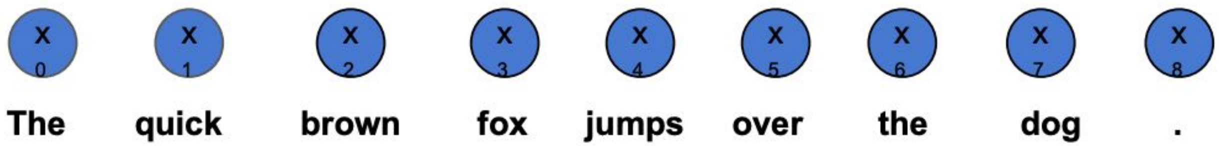
$$\mathbf{v}_0 = \mathbf{A}(\mathbf{x}_0, \mathbf{x}_1)$$

$$v_{0i} = A_i(\mathbf{x}_0, \mathbf{x}_1) = \sum_j (K_{0ij} x_{0j} + K_{1ij} x_{1j})$$

Вероятность принадлежности к классу

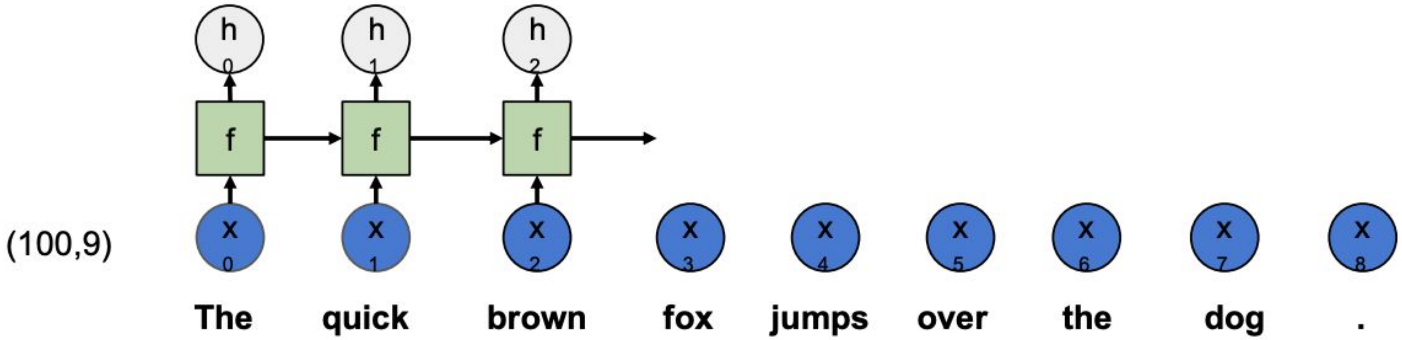


RNN for text



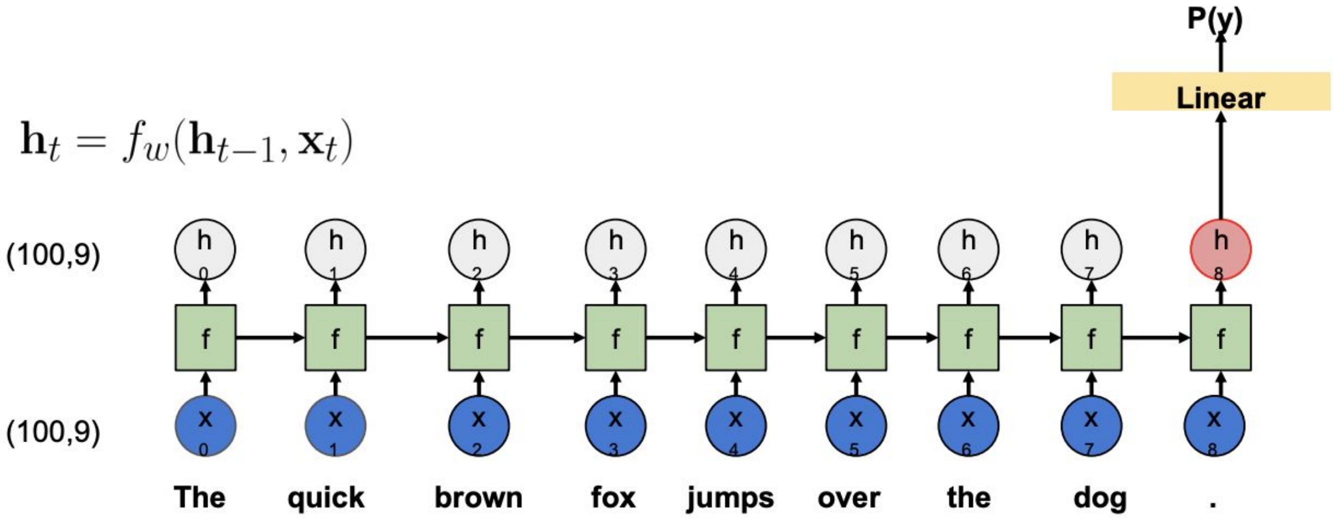
RNN for text

$$\mathbf{h}_t = f_w(\mathbf{h}_{t-1}, \mathbf{x}_t)$$



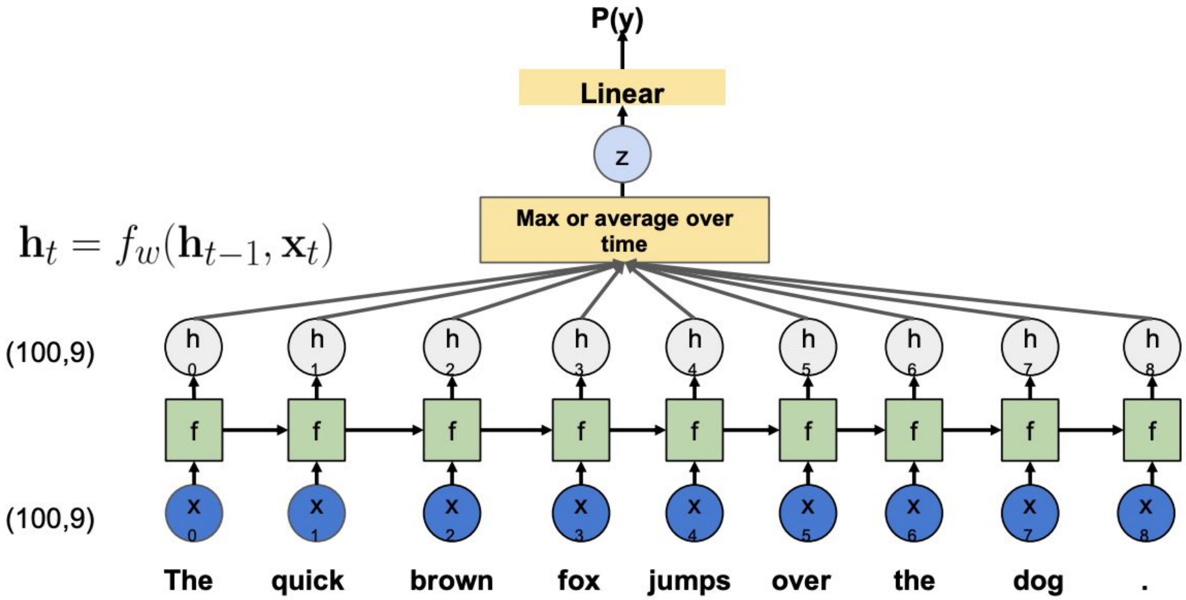
RNN for text

$$\mathbf{h}_t = f_w(\mathbf{h}_{t-1}, \mathbf{x}_t)$$

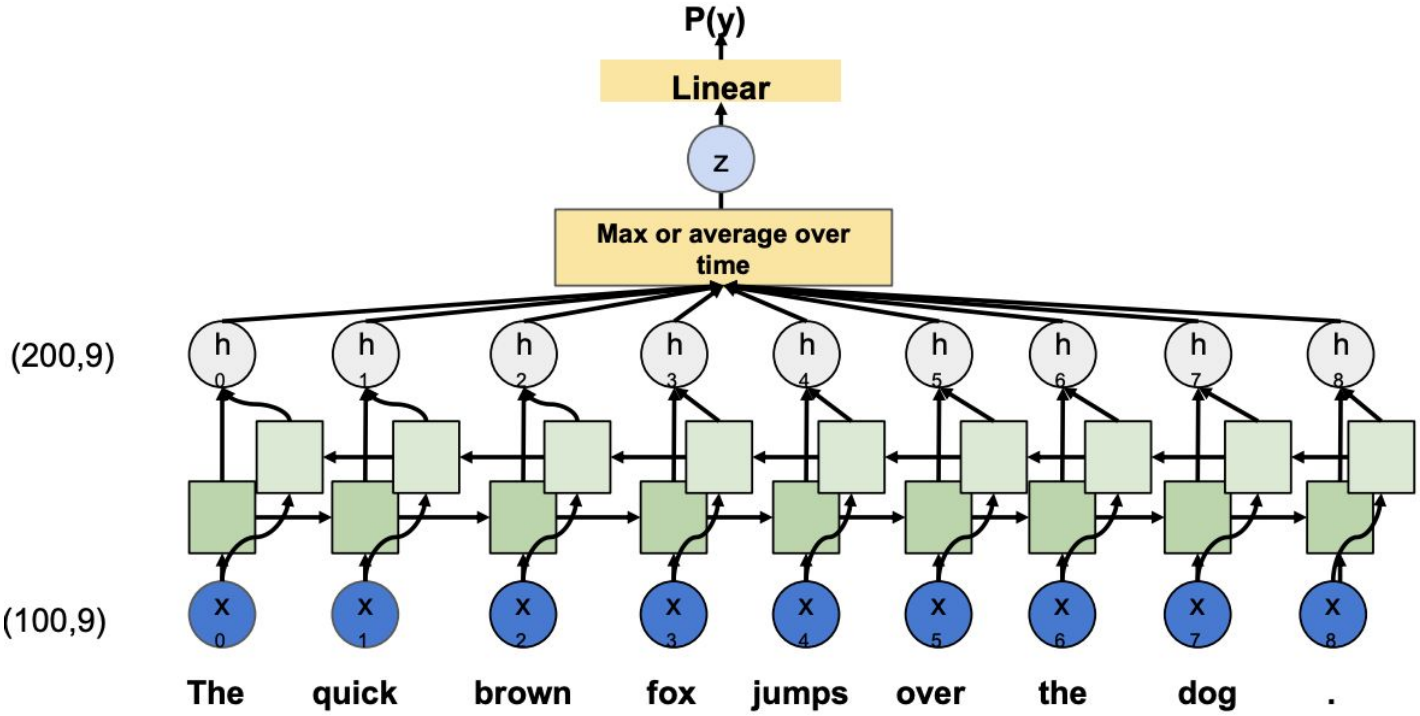


RNN for text

$$\mathbf{h}_t = f_W(\mathbf{h}_{t-1}, \mathbf{x}_t)$$



RNN for text



P(y)
Linear
z
Max or average over time
(200,9)
h0
h1
h2
h3
h4
h5
h6
h7
h8
(100,9)
x0
x1
x2
x3
x4
x5
x6
x7
x8
The
quick
brown
fox
jumps
over
the
dog
.

References

[1] https://arxiv.org/pdf/1506.02078.pdf , Karpathy et al, ICLR workshop, 2016
[2] https://github.com/yandexdataschool/nlp_course
[3] https://colah.github.io/posts/2015-08-Understanding-LSTMs/
[4] https://towardsdatascience.com/animated-rnn-lstm-and-gru-ef124d06cf45

References

[1] Blog posts introduction to pointer networks

[2] http://web.stanford.edu/class/cs224n/

[3] Mikolov et al, 2013, https://arxiv.org/pdf/1310.4546.pdf

[4] http://ruder.io/word-embeddings-1/

[5] http://aclweb.org/anthology/Q17-1010

[6] https://github.com/yandexdataschool/nlp_course

[7] https://papers.nips.cc/paper/5950-skip-thought-vectors.pdf

Main papers about Embeddings

- Distributed Representations of Words and Phrases and their Compositionality Mikolov et al., 2013 [arxiv]
- Efficient Estimation of Word Representations in Vector Space Mikolov et al., 2013 [arxiv]
- Distributed Representations of Sentences and Documents Quoc Le et al., 2014 [arxiv]
- GloVe: Global Vectors for Word Representation Pennington et al., 2014 [article]
- Enriching word vectors with subword information Bojanowski et al., 2016 [arxiv] (FastText)