

# Lecture 3: Word vectors & Recurrent Neural Networks

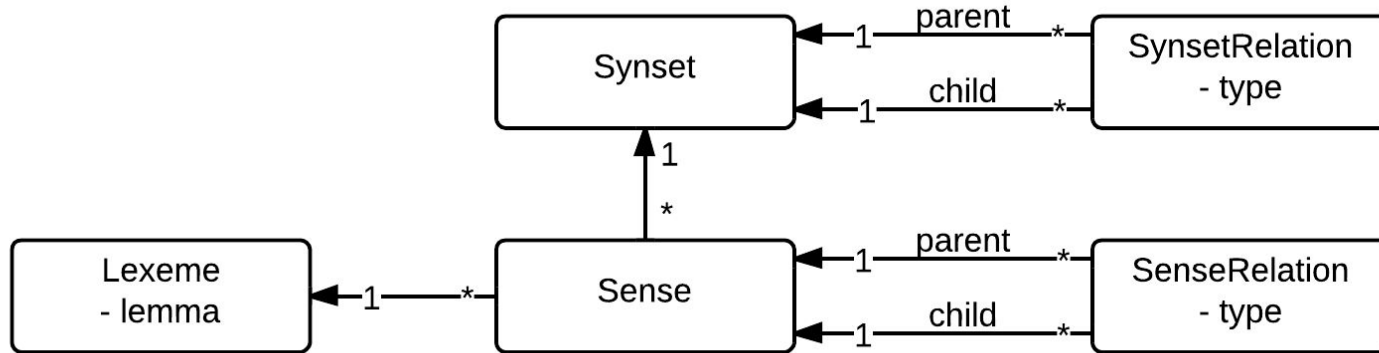
**Radoslav Neychev**

23.05.2020, Moscow, Russia

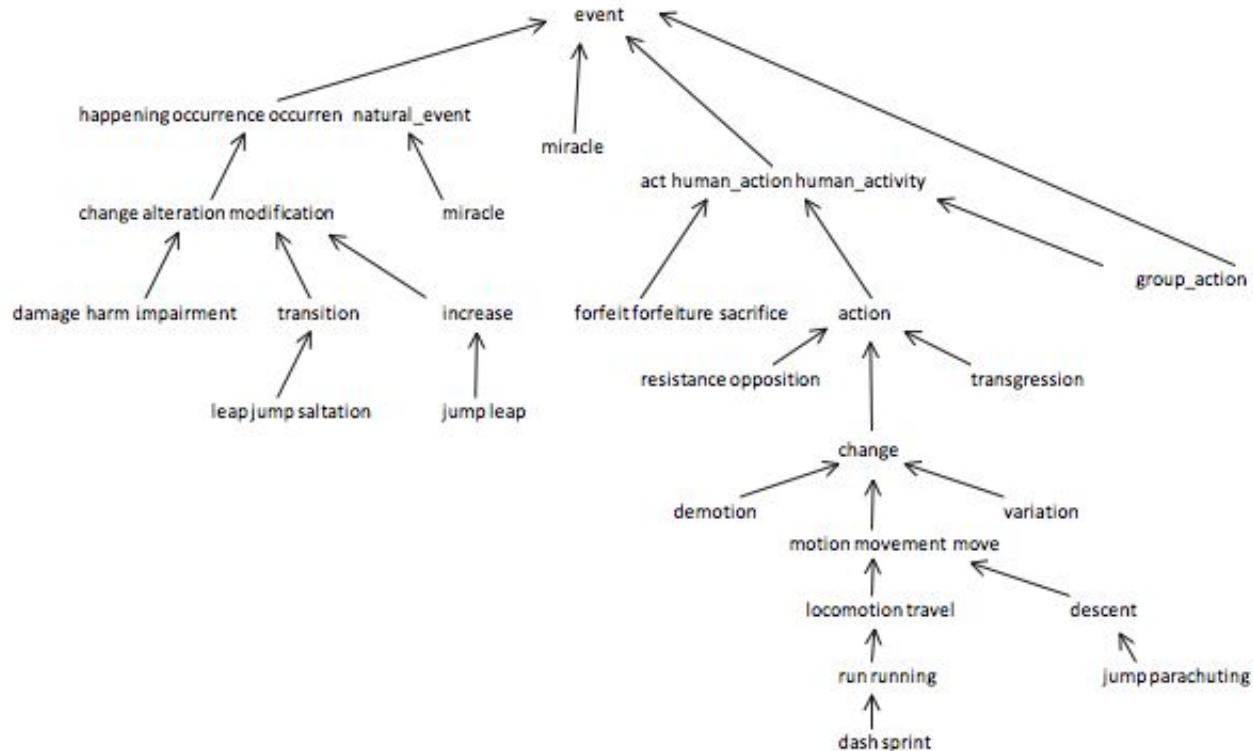
1. Discrete representations.
2. Matrix of co-occurrence.
3. Embeddings (GloVe, word2vec).
4. Examples.
5. Recurrent Neural Networks
6. Optional Discussion
7. Practice Session

# How to represent text in a computer?

Use a taxonomy like WordNet that has hypernyms (is-a) relationships and synonym sets



# How to represent text in a computer: WordNet



# Discrete representations: problems

- Missing new words
- Subjective
- Requires human labor to create and adapt
- Hard to compute accurate word similarity

# Discrete representations: one-hot encoding

"a"	"abbreviations"		"zoology"	"zoom"
1	0		0	0
0	1		0	1
0	0		0	0
⋮	⋮	⋮	⋮	⋮
0	0		0	0
0	0		1	0
0	0		0	1

The diagram shows the TF-IDF formula with several annotations in red text and arrows pointing to specific parts of the formula:

- Annotation 1:** "If word is repeated in the query, it's probably important" points to the term frequency in the query,  $tf_{w,Q}$ .
- Annotation 2:** "Repetitions of query words in the document → good" points to the term frequency in the document,  $tf_{w,D}$ .
- Annotation 3:** "Rare words more important" points to the inverse document frequency term,  $\log \frac{|C|}{df_w}$ .
- Annotation 4:** "Repetitions of same word less important than different words. Except in very long documents" points to the denominator of the IDF term,  $tf_{w,D} + \frac{k|D|}{avg|D|}$ .
- Annotation 5:** "The more query words we match, the better. Σ over the vocabulary" points to the summation symbol  $\sum_w$ .

$$s(Q,D) = \sum_w tf_{w,Q} \cdot \frac{tf_{w,D}}{tf_{w,D} + \frac{k|D|}{avg|D|}} \cdot \log \frac{|C|}{df_w}$$

TF - term frequency

IDF - Inverse Document Frequency

# TF-IDF: make it simple

$$\text{tf}(\text{"this"}, d_1) = \frac{1}{5} = 0.2$$

$$\text{tf}(\text{"this"}, d_2) = \frac{1}{7} \approx 0.14$$

$$\text{idf}(\text{"this"}, D) = \log\left(\frac{2}{2}\right) = 0$$



$$\text{tfidf}(\text{"this"}, d_1, D) = 0.2 \times 0 = 0$$

$$\text{tfidf}(\text{"this"}, d_2, D) = 0.14 \times 0 = 0$$



Word 'this' is not very  
informative

Document 1		Document 2	
Term	Term Count	Term	Term Count
this	1	this	1
is	1	is	1
a	2	another	2
sample	1	example	3



One of the most successful ideas of statistical NLP:

“You shall know a word by the company it keeps”

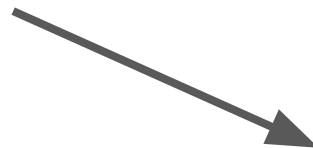
(J. R. Firth 1957: 11)

# Words cooccurrences

Finding N-grams in a text



Word-document  
cooccurrence matrix

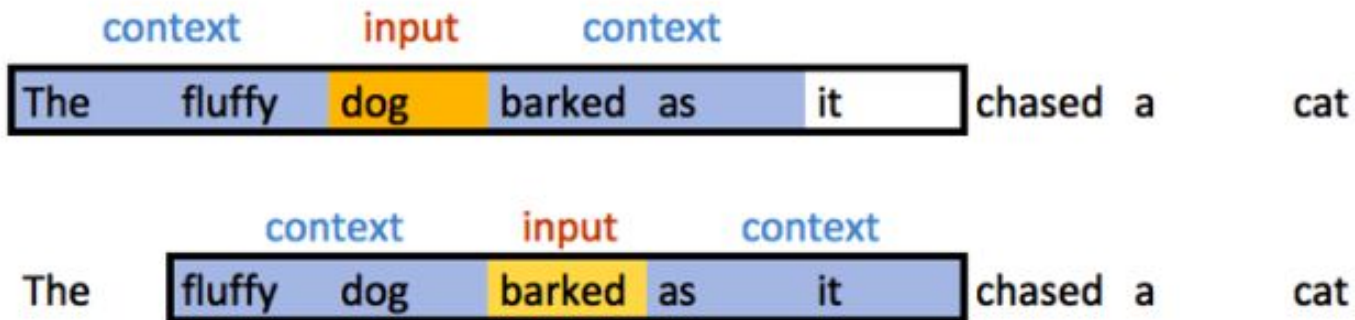


Window around  
each word

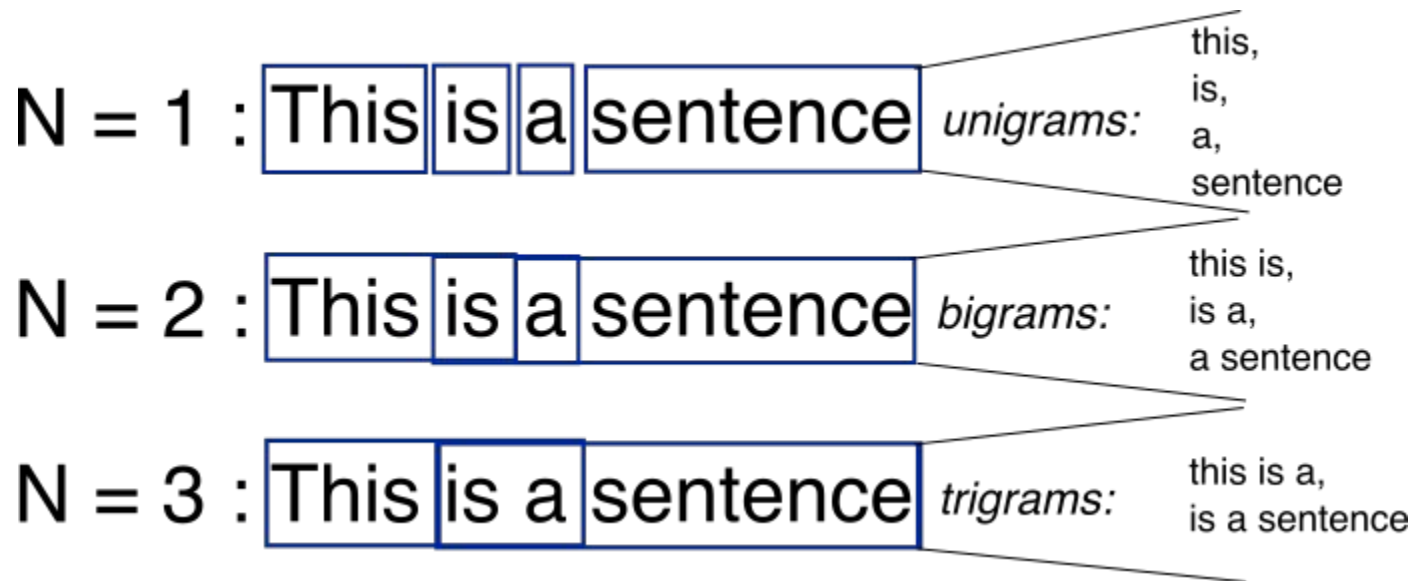
# Word-document cooccurrence matrix

$$X = \begin{matrix} & \begin{matrix} I & like & enjoy & deep & learning & NLP & flying & . \end{matrix} \\ \begin{matrix} I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{matrix} & \left[ \begin{array}{cccccccc} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{array} \right] \end{matrix}$$

# Words cooccurrences: sliding window



# Words cooccurrences: n-grams



# Cooccurrence vectors: problems

- Increase in size with vocabulary
- Very high dimensional: require a lot of storage
- Subsequent classification models have sparsity issues



Models are less robust

# Reducing dimensionality: SVD of cooccurrence matrix

Item x subject matrix  
(ISM)

	S1	S2	S3	S4	S5
dog	1	1	1	1	1
cat	1	1	0	1	0
cow	0	0	1	0	1
lion	0	0	1	1	0
tiger	1	1	0	0	1

Singular decomposition  
analysis (SVD)

$$C_{m \times n} = U_{m \times r} \times \Sigma_{r \times r} \times V'_{r \times n}$$

Item vectors      Singular values      Subject vectors

Reducing dimensions  
from  $r$  to  $k$

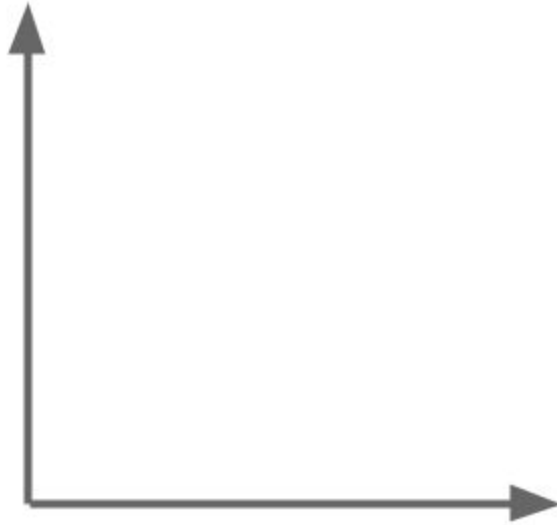


$$\tilde{C}_{m \times n} = U_{m \times k} \times \Sigma_{k \times k} \times V'_{k \times n}$$

Item vectors      Singular values      Subject vectors

# Embeddings: intuition

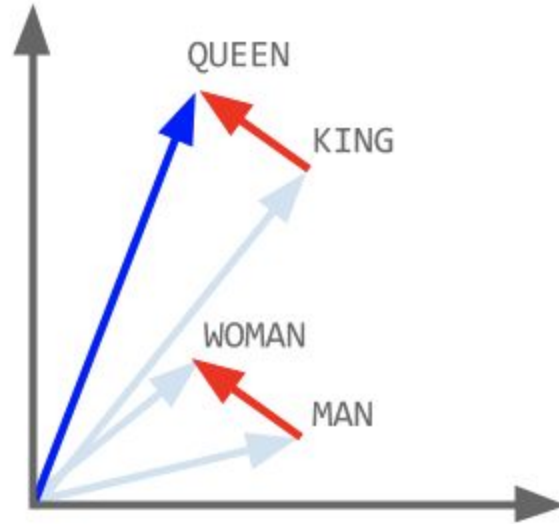
What is  $\text{king} - \text{man} + \text{woman}$ ?





# Embeddings: intuition

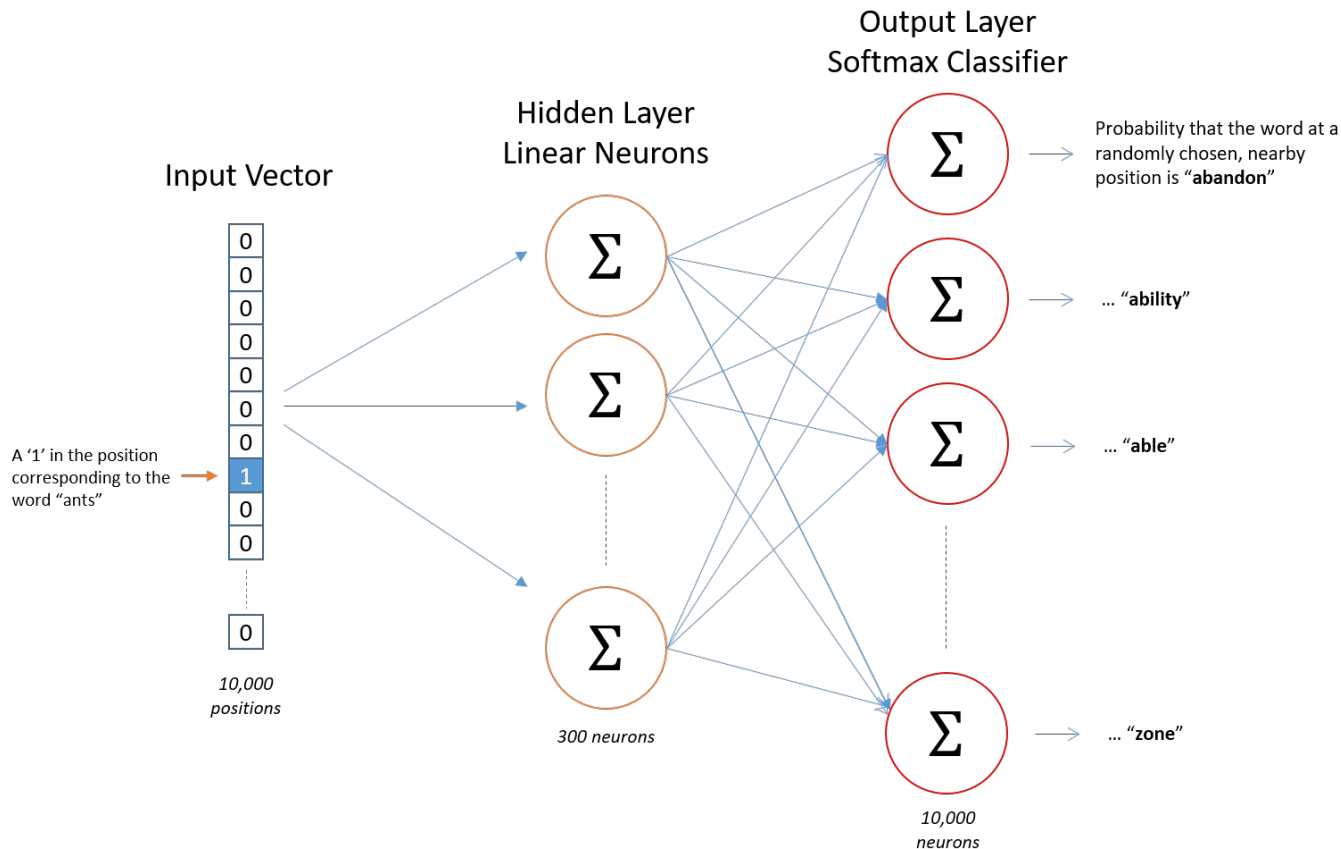
So  $\text{king} - \text{man} + \text{woman} = \text{queen!}$



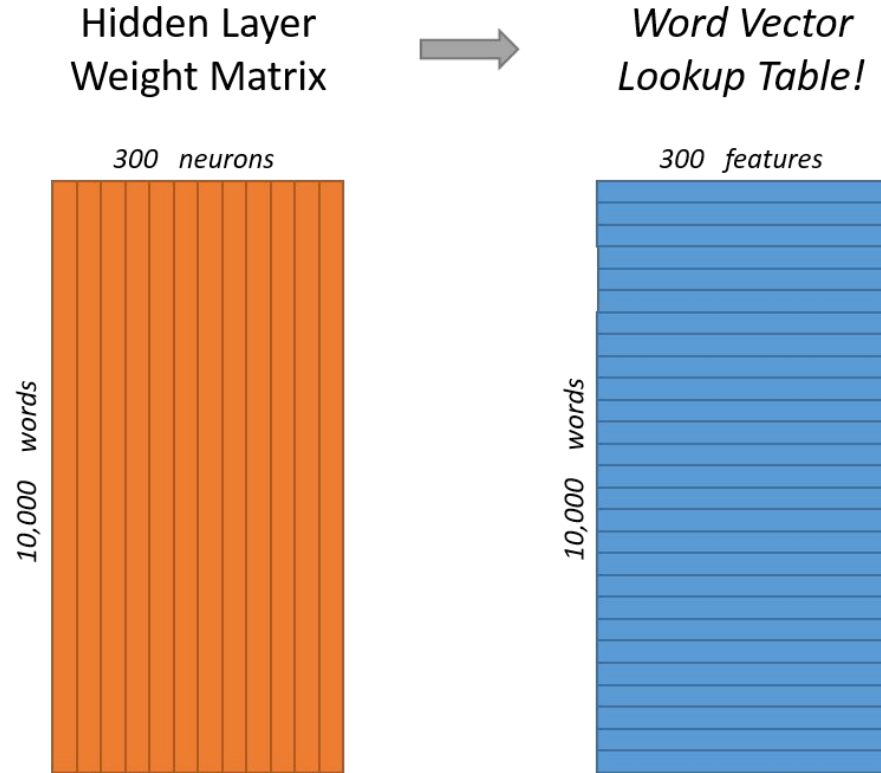
# Embeddings: word2vec

Source Text	Training Samples					
<table><tr><td>The</td><td>quick</td><td>brown</td></tr></table> fox jumps over the lazy dog. ➡	The	quick	brown	(the, quick) (the, brown)		
The	quick	brown				
The <table><tr><td>quick</td><td>brown</td><td>fox</td></tr></table> jumps over the lazy dog. ➡	quick	brown	fox	(quick, the) (quick, brown) (quick, fox)		
quick	brown	fox				
The quick <table><tr><td>brown</td><td>fox</td><td>jumps</td></tr></table> over the lazy dog. ➡	brown	fox	jumps	(brown, the) (brown, quick) (brown, fox) (brown, jumps)		
brown	fox	jumps				
The <table><tr><td>quick</td><td>brown</td><td>fox</td><td>jumps</td><td>over</td></tr></table> the lazy dog. ➡	quick	brown	fox	jumps	over	(fox, quick) (fox, brown) (fox, jumps) (fox, over)
quick	brown	fox	jumps	over		

# Embeddings: word2vec



# Embeddings: word2vec



# Embeddings: word2vec

- Word vectors with 300 components
- Vocabulary of 10,000 words.
- Weight matrix with  $300 \times 10,000 = 3$  million weights each!

Training is too long and computationally expensive

How to fix this?

## Basic approaches:

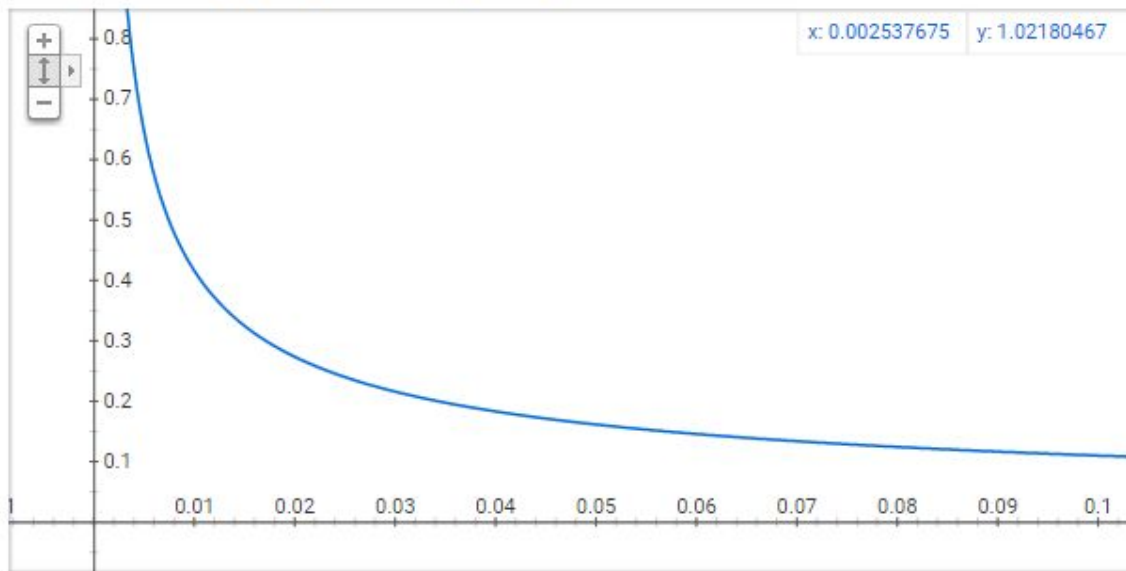
1. Treating common word pairs or phrases as single “words” in their model.
2. Subsampling frequent words to decrease the number of training examples.
3. Modifying the optimization objective with a technique they called “Negative Sampling”, which causes each training sample to update only a small percentage of the model’s weights.

# Embeddings: word2vec

Subsampling frequent words.

$w_i$  is the word,  $z(w_i)$  is the fraction of this word in the whole

Graph for  $(\sqrt{x/0.001}+1)*0.001/x$



$P(w_i)$  is the probability of *keeping* the word:

$$P(w_i) = \left( \sqrt{\frac{z(w_i)}{0.001}} + 1 \right) \cdot \frac{0.001}{z(w_i)}$$

Source: <http://mccormickml.com/2017/01/11/word2vec-tutorial-part-2-negative-sampling/>

# Embeddings: negative sampling

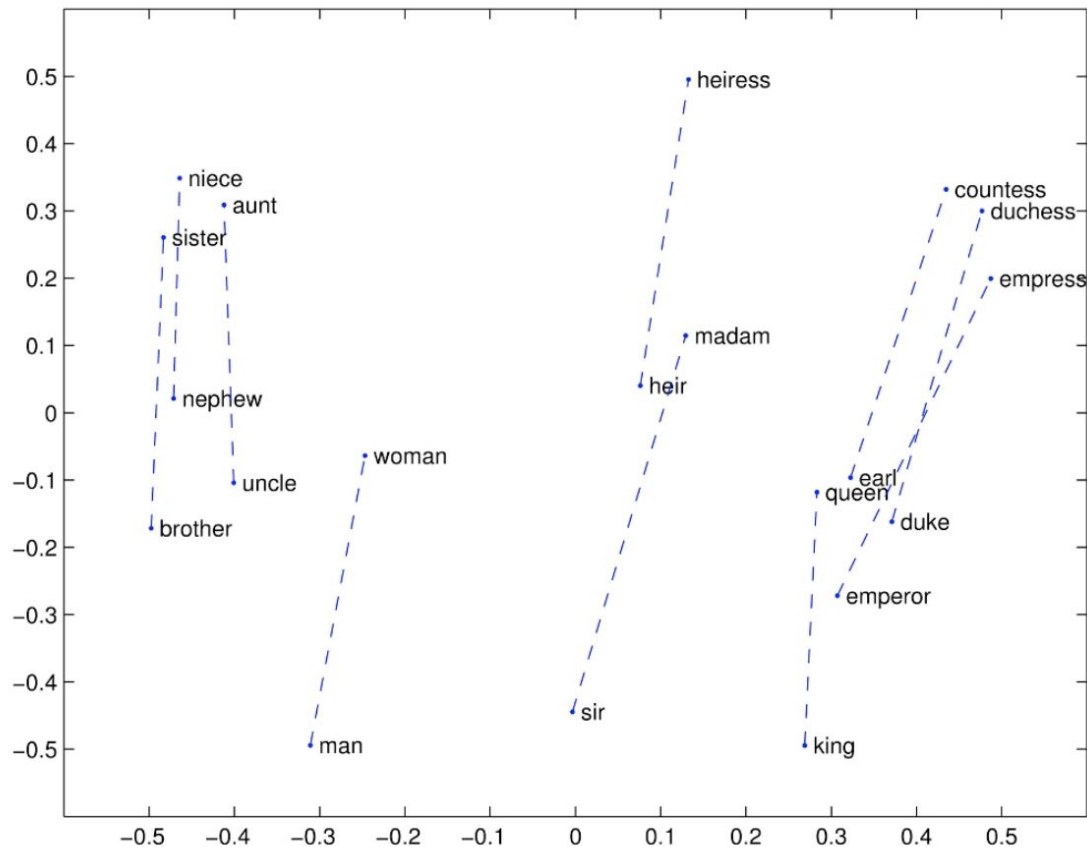
Negative Sampling idea: only few words error is computed. All other words has zero error, so no updates by the backprop mechanism.

More frequent words are selected to be negative samples more often. The probability for a selecting a word is just it's weight divided by the sum of weights for all words.

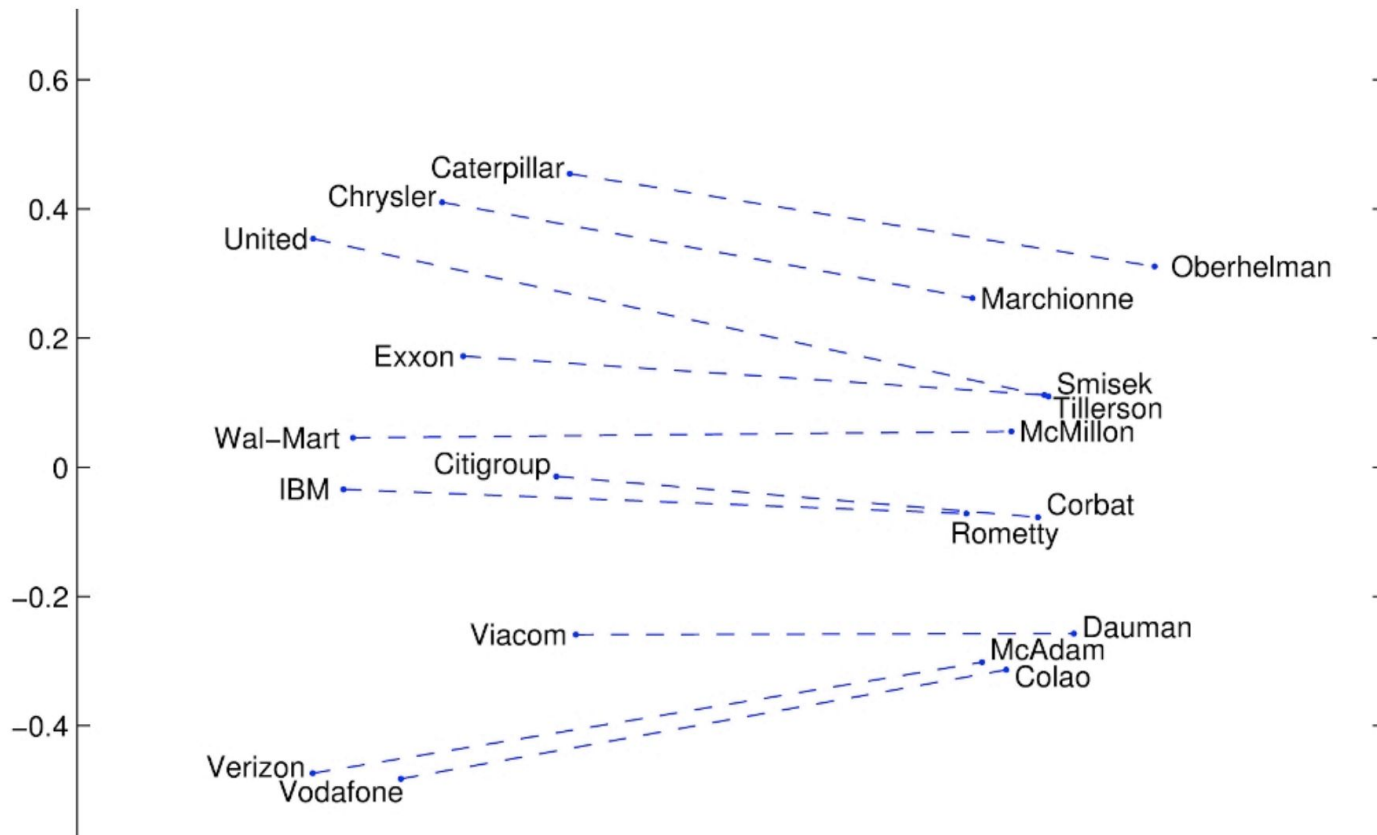
$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^n (f(w_j)^{3/4})}$$



# GloVe Visualizations



# GloVe Visualizations: Company - CEO



Word vectors are simply vectors of numbers that represent the meaning of a word

Approaches:

- One-hot encoding
- Bag-of-words models
- Counts of word / context co-occurrences
- TF-IDF
- Predictions of context given word (skip-gram neural network models, e.g. word2vec)

# RNNs generating...

## Shakespeare

PANDARUS:  
Alas, I think he shall be come approached and the day  
When little strain would be attain'd into being never fed,  
And who is but a chain and subjects of his death,  
I should not sleep.

Second Senator:  
They are away this miseries, produced upon my soul,  
Breaking and strongly should be buried, when I perish  
The earth and thoughts of many states.

DUKE VINCENTIO:  
Well, your wit is in the care of side and that.

Second Lord:  
They would be ruled after this chamber, and  
my fair nues begun out of the fact, to be conveyed,  
Whose noble souls I'll have the heart of the wars.

Clown:  
Come, sir, I will make did behold your worship.

VIOLA:  
I'll drink it.

## Algebraic Geometry (Latex)

*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*

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

*Proof.* This is an algebraic space with the composition of sheaves  $\mathcal{F}$  on  $X_{\text{étale}}$  we have

$$\mathcal{O}_X(\mathcal{F}) = \{\text{morph}_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

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

**Lemma 0.2.** *This is an integer  $\mathbb{Z}$  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  $U \subset X$  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*

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

*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. □

## Linux kernel (source code)

```
/*  
 * If this error is set, we will need anything right after that BSD.  
 */  
  
static void action_new_function(struct s_stat_info *wb)  
{  
    unsigned long flags;  
    int lel_idx_bit = e->add, *sys & -((unsigned long) *FIRST_COMPAT);  
    buf[0] = 0xffffffff & (bit << 4);  
    min(inc, slist->bytes);  
    printk(KERN_WARNING "Memory allocated %02x/%02x, "  
        "original MLL instead\n"),  
        min(min(multi_run - s->len, max) * num_data_in),  
        frame_pos, sz + first_seg);  
    div_u64_w(val, inb_p);  
    spin_unlock(&disk->queue_lock);  
    mutex_unlock(&s->sock->mutex);  
    mutex_unlock(&func->mutex);  
    return disassemble(info->pending_bh);  
}
```

*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  $X_{\text{étale}}$  we have

$$\mathcal{O}_X(\mathcal{F}) = \{ \text{morph}_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F}) \}$$

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

**Lemma 0.2.** *This is an integer  $Z$  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*

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

*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

$$\begin{array}{ccc} S & \xrightarrow{\quad} & \\ \downarrow & & \downarrow \\ \xi & \xrightarrow{\quad} & \mathcal{O}_{X'} \\ \text{gor}_s & \uparrow & \searrow \\ & \alpha' & \\ & \downarrow & \\ & \alpha' & \rightarrow \alpha \end{array} \quad \begin{array}{c} X \\ \downarrow \\ \text{Mor}_{\text{Sets}}(\mathcal{O}_{X_{X/S}}, \mathcal{G}) \end{array}$$

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  $X = \text{Spec}(R)$  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

$$\mathcal{O}_{X,x} \rightarrow \mathcal{F}_x \rightarrow -1(\mathcal{O}_{X_{\text{étale}}}) \rightarrow \mathcal{O}_{X_t}^{-1} \mathcal{O}_{X_\lambda}(\mathcal{O}_{X_q}^v)$$

is an isomorphism of covering of  $\mathcal{O}_{X_t}$ . 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.

```

#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/setew.h>
#include <asm/pgproto.h>

#define REG_PG    vesa_slot_addr_pack
#define PFM_NOCOMP AFSR(0, load)
#define STACK_DDR(type)      (func)

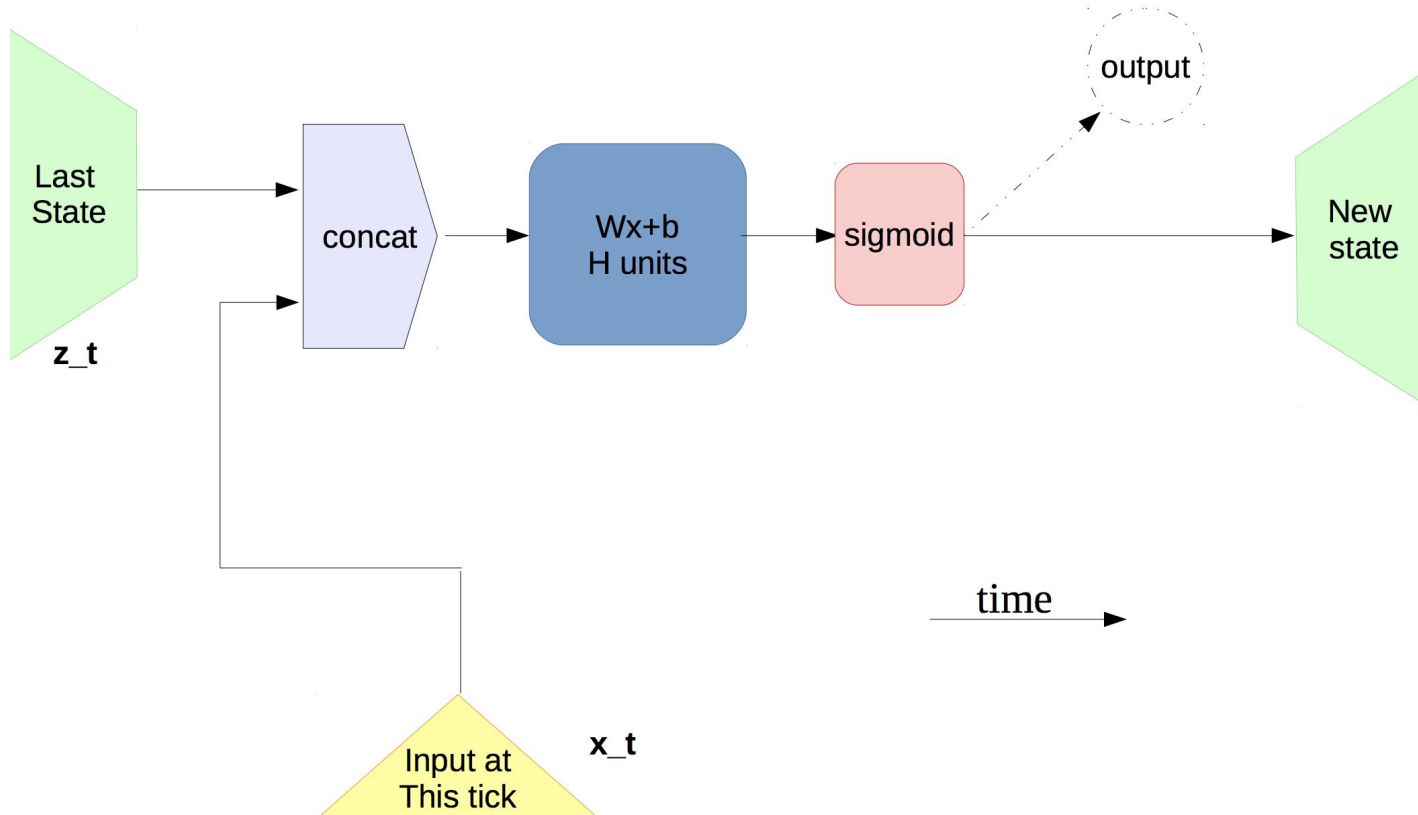
#define SWAP_ALLOCATE(nr)      (e)
#define emulate_sigs()  arch_get_unaligned_child()
#define access_rw(TST)  asm volatile("movd %%esp, %0, %3" : : "r" (0)); \
    if (__type & DO_READ)

static void stat_PC_SEC __read_mostly offsetof(struct seq_argsqueue, \
    pC>[1]);

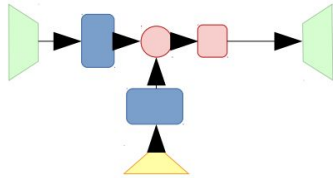
static void
os_prefix(unsigned long sys)
{
#ifdef CONFIG_PREEMPT
    PUT_PARAM_RAID(2, sel) = get_state_state();
    set_pid_sum((unsigned long)state, current_state_str(),
        (unsigned long)-1->lr_full; low;
}

```

# Recurrent neural network



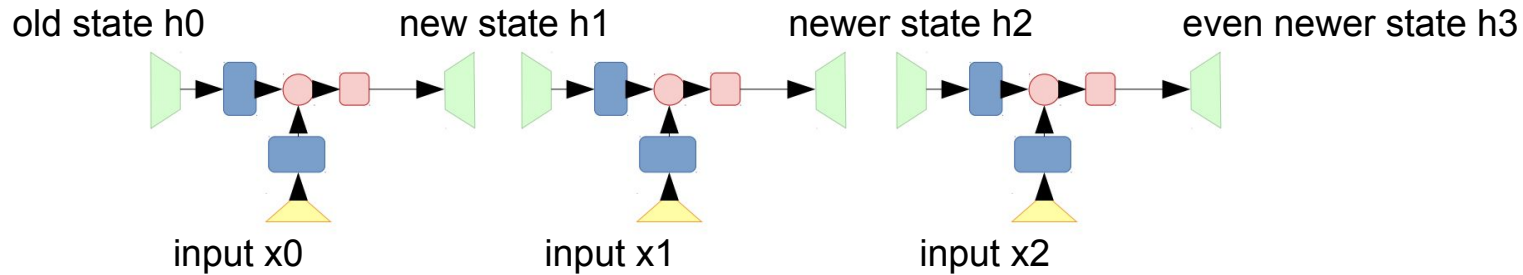
# Recurrent neural network



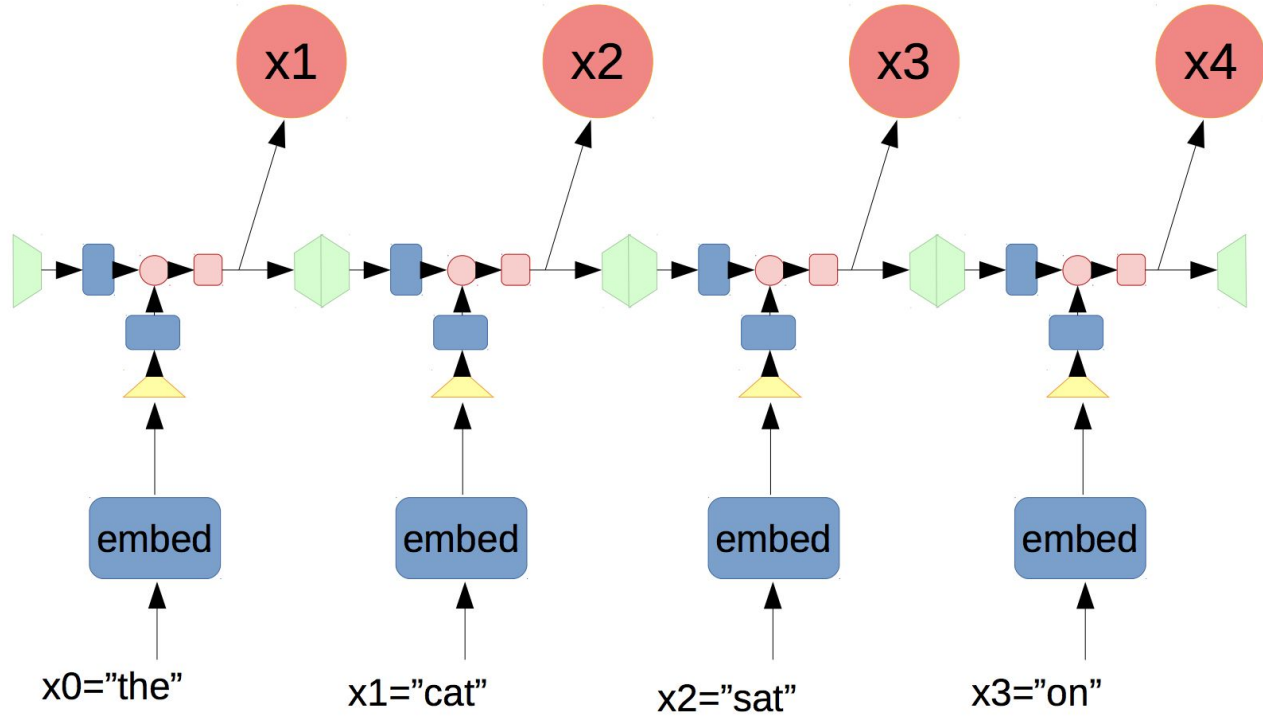


# Recurrent neural network

We use same weight matrices for all steps



# Recurrent neural network



# Recurrent neural network

Now with formulas

$$h_0 = \bar{0}$$

$$h_1 = \sigma(\langle W_{\text{hid}}[h_0, x_0] \rangle + b)$$

$$h_2 = \sigma(\langle W_{\text{hid}}[h_1, x_1] \rangle + b) = \sigma(\langle W_{\text{hid}}[\sigma(\langle W_{\text{hid}}[h_0, x_0] \rangle + b), x_1] \rangle + b)$$

$$h_{i+1} = \sigma(\langle W_{\text{hid}}[h_i, x_i] \rangle + b)$$

$$P(x_{i+1}) = \text{softmax}(\langle W_{\text{out}}, h_i \rangle + b_{\text{out}})$$

# RNNs generating...

## Shakespeare

PANDARUS:  
Alas, I think he shall be come approached and the day  
When little strain would be attain'd into being never fed,  
And who is but a chain and subjects of his death,  
I should not sleep.

Second Senator:  
They are away this miseries, produced upon my soul,  
Breaking and strongly should be buried, when I perish  
The earth and thoughts of many states.

DUKE VINCENTIO:  
Well, your wit is in the care of side and that.

Second Lord:  
They would be ruled after this chamber, and  
my fair nues begun out of the fact, to be conveyed,  
Whose noble souls I'll have the heart of the wars.

Clown:  
Come, sir, I will make did behold your worship.

VIOLA:  
I'll drink it.

## Algebraic Geometry (Latex)

*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*

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

*Proof.* This is an algebraic space with the composition of sheaves  $\mathcal{F}$  on  $X_{\text{étale}}$  we have

$$\mathcal{O}_X(\mathcal{F}) = \{\text{morph}_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

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

**Lemma 0.2.** *This is an integer  $\mathbb{Z}$  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  $\mathcal{U} \subset X$  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*

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

*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

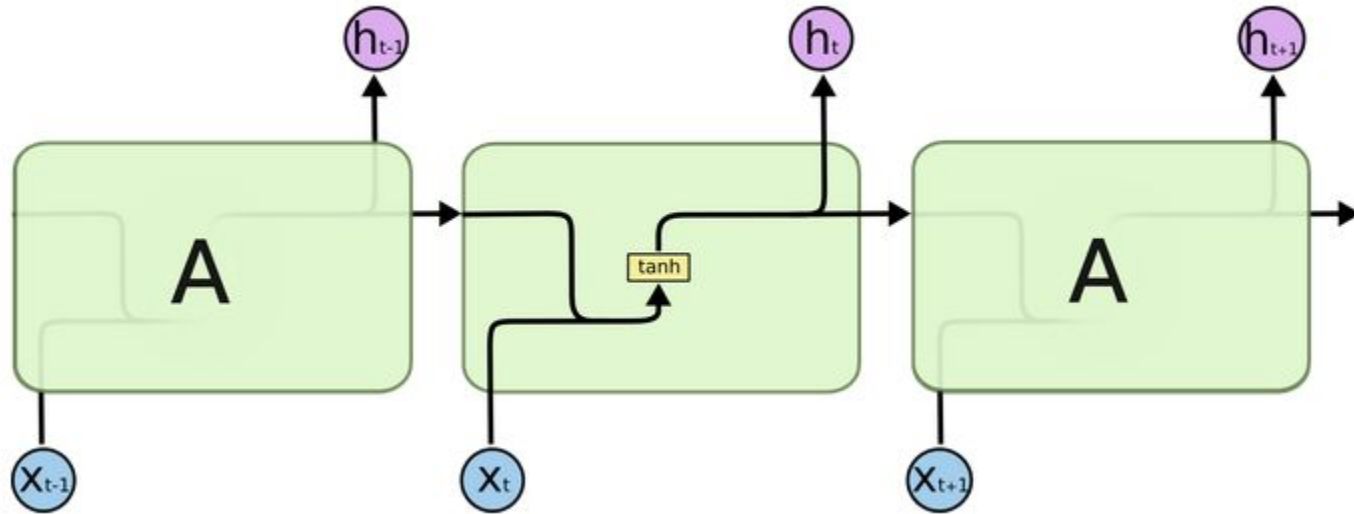
- (1)  $\mathcal{F}$  is an algebraic space over  $S$ .
- (2) If  $X$  is an affine open covering.

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

## Linux kernel (source code)

```
/*  
 * If this error is set, we will need anything right after that BSD.  
 */  
  
static void action_new_function(struct s_stat_info *wb)  
{  
    unsigned long flags;  
    int lel_idx_bit = e->add, *sys & -((unsigned long) *FIRST_COMPAT);  
    buf[0] = 0xffffffff & (bit << 4);  
    min(inc, slist->bytes);  
    printk(KERN_WARNING "Memory allocated %02x/%02x, "  
           "original MLL instead\n"),  
           min(min(multi_run - s->len, max) * num_data_in),  
           frame_pos, sz + first_seg);  
    div_u64_w(val, inb_p);  
    spin_unlock(&disk->queue_lock);  
    mutex_unlock(&s->sock->mutex);  
    mutex_unlock(&func->mutex);  
    return disassemble(info->pending_bh);  
}
```

# Vanilla RNN

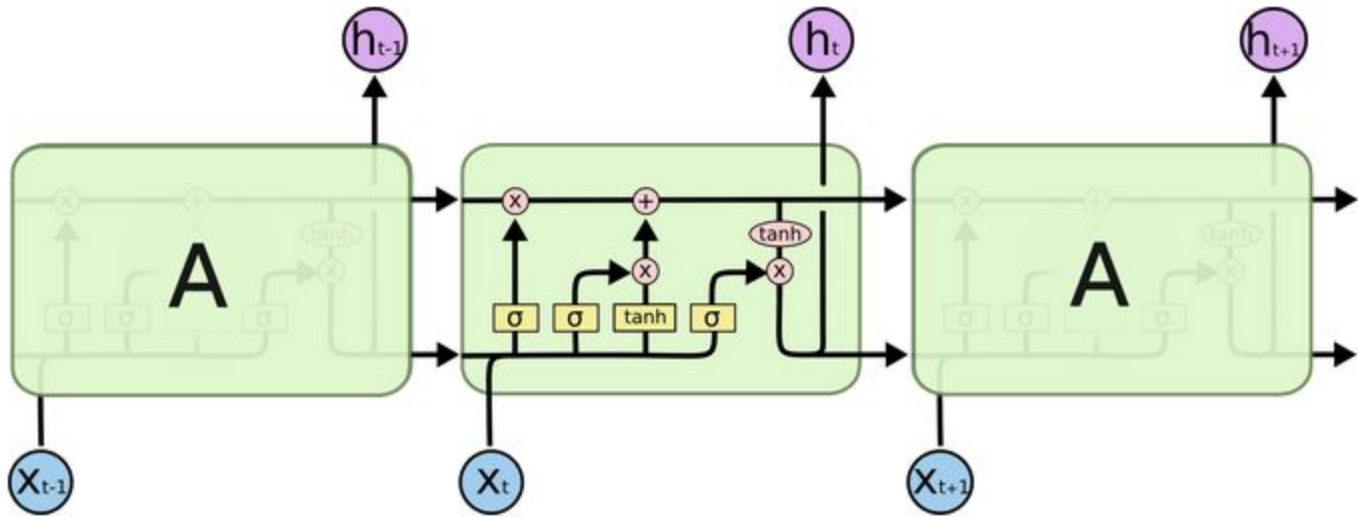


That's all. Feel free to ask any questions.

Remember: *a well-defined problem is halfway to being solved*

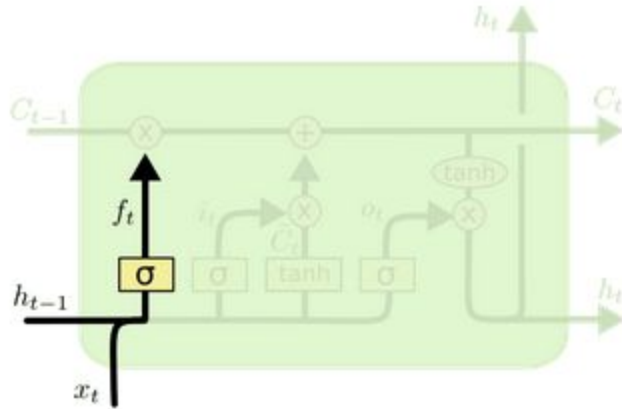
# Optional

# LSTM



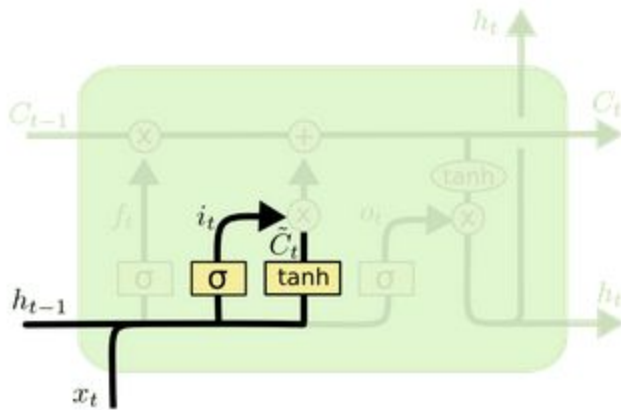


# LSTM: quick overview



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

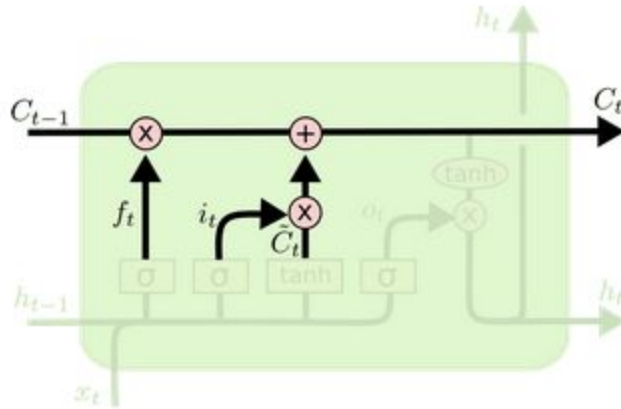
# LSTM: quick overview



$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

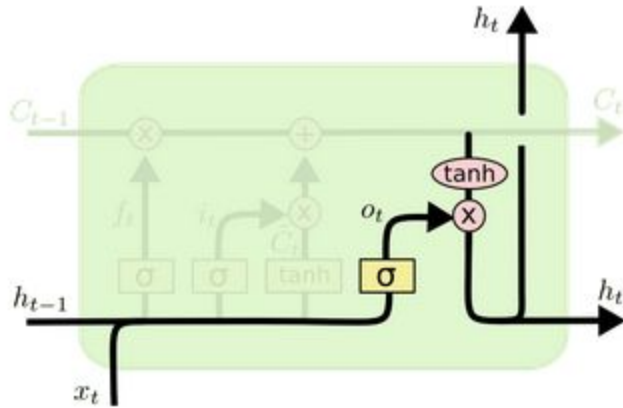
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

# LSTM: quick overview



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

# LSTM: quick overview



$$o_t = \sigma (W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh (C_t)$$

- Do not forget about dropout. It really rocks.
- Other regularization approaches are welcome as well (see previous lectures).
- Combining RNN and CNN worlds? *Coming soon*