

## 13 - SEQUENCE MODELS

### Task with sequence data

|                            |   |   |   |
|----------------------------|---|---|---|
| Speech recognition         |   | → | "The quick brown fox jumped over the lazy dog." |
| Music generation           | ∅   | → |   |
| Sentiment classification   | "There is nothing to like in this movie."     | → | ★☆☆☆☆   |
| DNA sequence analysis      | AGCCCCCTGTGAGGAAC TAG                         | → | AGCCCCTGTGAGGAAC TAG                            |
| Machine translation        | Voulez-vous chanter avec moi?                 | → | Do you want to sing with me?                    |
| Video activity recognition |   | → | Running   |
| Named-entity recognition   | Yesterday, Harry Potter met Hermione Granger. | → | Yesterday, Harry Potter met Hermione Granger.   |

### Notation

Abbiamo una frase di **input x** e indicizziamo gli elementi e affrontiamo il problema con un approccio supervised learning avendo quindi un output **y** che ci dice quali sono le cose rilevanti.

Ogni singola parola la descriviamo con una  $x^{<t>}$  a cui viene associato un certo time step, allo stesso modo ogni output lo descriviamo con  $y^{<t>}$ .  $T_x$  e  $T_y$  non devono avere necessariamente la stessa dimensione.

**x:** Harry Potter and Hermione Granger invented a new spell.

|    | $x^{<1>}$ | $x^{<2>}$ | $x^{<t>}$          | ... | $x^{<9>}$ |         |
|----|-----------|-----------|--------------------|-----|-----------|---------|
|    |           |           | Size of x: $T_x=9$ |     |           |         |
| y: | 1         | 1         | 0                  | 1   | 1         | 0 0 0 0 |

|  | $y^{<1>}$ | $y^{<2>}$ | ... | $y^{<9>}$          |  |
|--|-----------|-----------|-----|--------------------|--|
|  |           |           |     | Size of y: $T_y=9$ |  |

$T_x$  not necessarily equal to  $T_y$

$t^{\text{th}}$  element of the  $i^{\text{th}}$  training example:  $x^{(i)<t>}$

Size of x for the  $i^{\text{th}}$  training example:  $T_x^{(i)}$  (same for y,  $T_y^{(i)}$ )

### Representing words

Si può pensare di creare un vocabolario e ogni parola la mappiamo con una one-hot representation che indica la parola del vocabolario che ha rilevanza. (one-hot encoding: vettore conterrà tutti 0 tranne nella posizione corrispondente alla nostra parola)

**x:** Harry Potter and Hermione Granger invented a new spell.

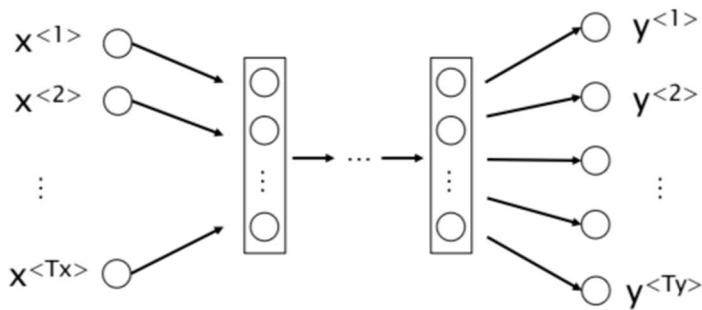
| <u>Vocabulary</u>                 | $x^{<1>}$ | $x^{<2>}$ | $x^{<t>}$ | ... | $x^{<9>}$ |   |
|-----------------------------------|-----------|-----------|-----------|-----|-----------|---|
| aaron                             | 1         | 2         | 0         | 0   | ...       | → Tensore bidimensionale: un asse rappresenta la dimensione del tempo, l'altro asse rappresenta le feature numeriche del nostro campione. |
| ...                               | ...       | ...       | ...       | ... | ...       |   |
| and                               | 367       | ...       | ...       | ... | ...       |   |
| ...                               | ...       | ...       | ...       | ... | ...       |   |
| harry                             | 4.075     | ...       | 0         | ... | ...       |   |
| ...                               | ...       | ...       | 1         | ... | ...       |   |
| potter                            | 6.830     | ...       | 0         | 0   | ...       |   |
| ...                               | ...       | ...       | ...       | ... | ...       |   |
| zulu                              | 10.000    | ...       | 0         | 0   | ...       |   |
| <b>&lt;UNK&gt;</b><br>for unknown |           |           |           |     |           |   |

Se sono presenti parole che non ci sono nel dizionario, queste si denotano come UNK; ci sono altri token per indicare alcune parole speciali. Parole che hanno una relazione, solitamente vicine.

Soltanamente si effettuano operazioni di manipolazione in modo tale da non avere differenze tra singolare/plurale, etc.

## Architetture usate:

### Using a “standard” neural network



Layer fully-connected e nell’ultimo layer, tante uscite quanti sono i nostri simboli. Ognuna di queste uscite sarà un vettore corrispondente all’elemento della sequenza di uscita o un valore tra 0-1 che mi rappresenta la probabilità che l’elemento  $t$  sia una entity dell’embedding oppure no.

### Problemi:

- Input/output potrebbero essere di lunghezze diverse per i vari esempi
  - o Posso determinare lunghezza massima, se non la raggiungo padding (non molto elegante perché devo fissare lunghezza massima e per tutte le stringhe di lunghezza inferiore sto sprecando tanto spazio e la rete deve capire come gestirle)
- Non vengono condivise le features imparate in porzioni diverse del testo
- Non ha un meccanismo di memoria, ogni parola processata in maniera indipendente dalle altre.

Per ovviare a questi problemi sono state proposte le seguenti architetture:

### Recurrent Neural Network (RNN)

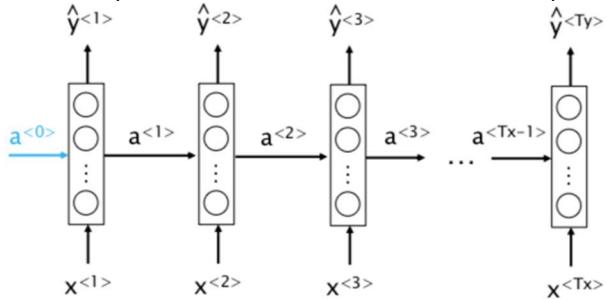
Prendo un layer di una rete standard a cui do in pasto una parola ottenendo un output; così per le altre parole.

Le RNN portano alcune informazioni da un passo temporale all’altro.

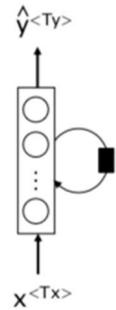
Quello che viene portato avanti sono le attivazioni del livello che vengono chiamate gli hidden state.

Quindi lo si fa per tutti i time steps e tipicamente si inizializza il modello con un vettore di input a 0 o con qualcosa di casuale.

La visione qui sotto è chiamata unrolled ed è usata per capire meglio la visualizzazione.

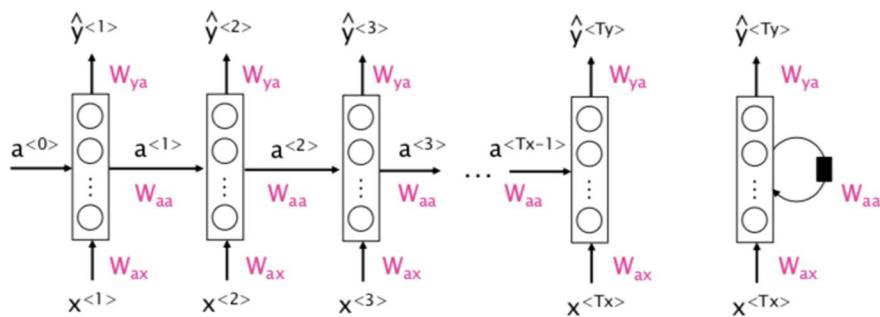


Le RNN possono anche essere rappresentate come un unico livello che ha la ricorrenza su sé stesso.



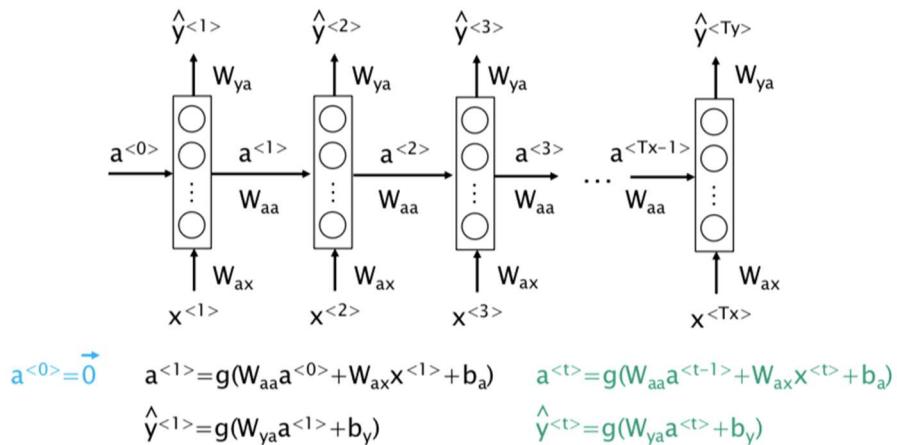
Per passare dagli input alle attivazioni si moltiplicano dei pesi e si somma un bias.

- La matrice dei pesi di input  $W_{ax}$  viene moltiplicata a tutti i time steps
  - o Per predizioni
  - o Uguale per tutti i timestamp
  - o Non dipende dalla lunghezza della rete ma dalla dimensione dell’embedding e dalla complessità
- una matrice dei pesi  $W_{aa}$  per le attivazioni
- una matrice dei pesi  $W_{ya}$  per gli output.



Nota:  $y \rightarrow$  output in base a cosa ho visto

## Forward propagation



Molto spesso si va semplificare questo modello staccando insieme le  $W_a$  che moltiplicano le attivazioni e gli input e nella parte di uscita si butta il pedice "a" e "x" come nell'immagine seguente.

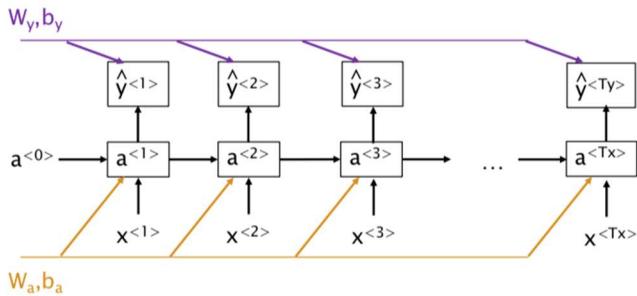
Simplified notation

$$\begin{cases} a^{<t>} = g(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \\ \hat{y}^{<t>} = g(W_{ya}a^{<t>} + b_y) \\ a^{<t>} = g(W_a[a^{<t-1>}, x^{<t>}] + b_a) \\ \hat{y}^{<t>} = g(W_ya^{<t>} + b_y) \end{cases}$$

## Backward propagation

Abbiamo bisogno di propagare le informazioni all'indietro per calcolare i gradienti dei parametri per fare il passo di update ad esempio usando il gradient descend.

Il computation graph per la RNN può essere rappresentato così.



Per addestrare questa rete serve una Loss e uso quello di una Logistic Regression per ogni time step.

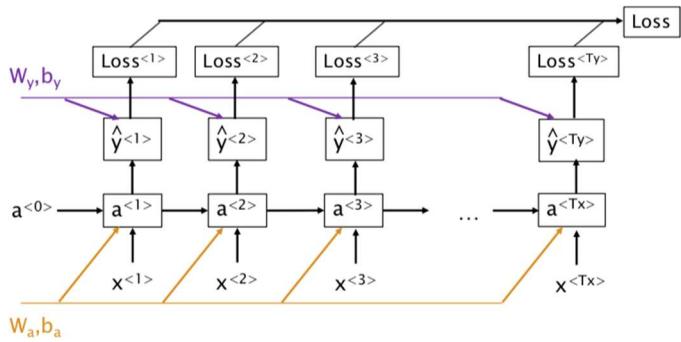
$$\text{Loss}^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -y^{<t>} \log(\hat{y}^{<t>}) - (1-y^{<t>}) \log(1-\hat{y}^{<t>})$$

Mettendole tutte insieme ottengo il Loss complessivo

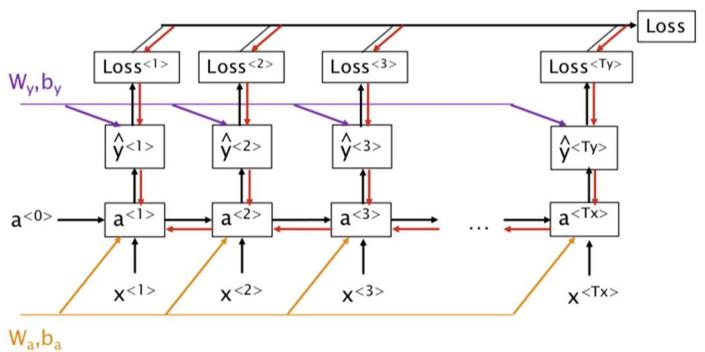
$$J(\hat{y}, y) = \sum_{t=1}^{T_y} \text{Loss}^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

Let me kn

Quindi il risultato finale è il seguente



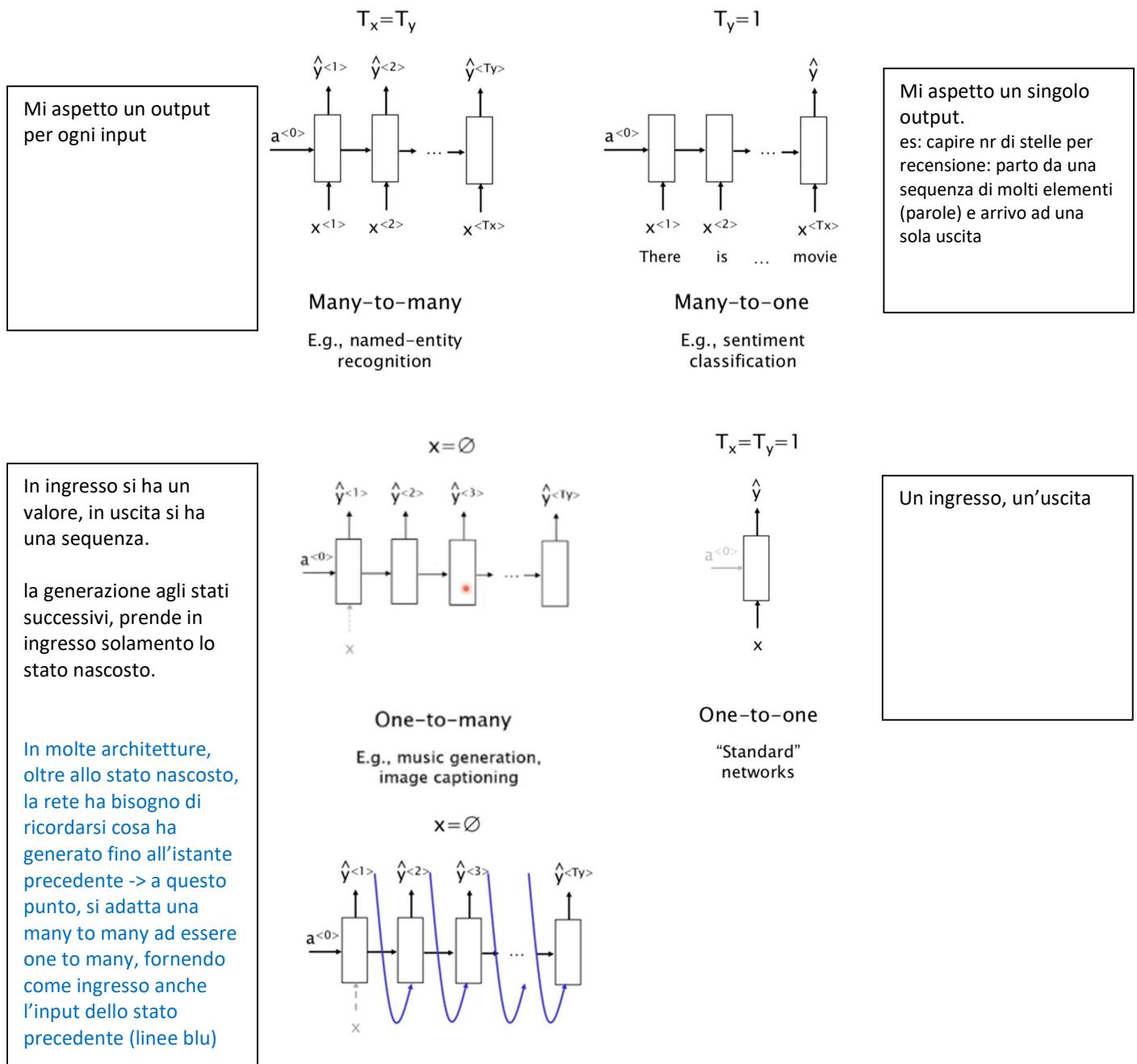
Da qui si fa il backpropagation "through time" che risulta essere molto lento nella computazione. (non posso parallelizzare)



## Architetture e esempi di applicazioni

### Architetture RNN

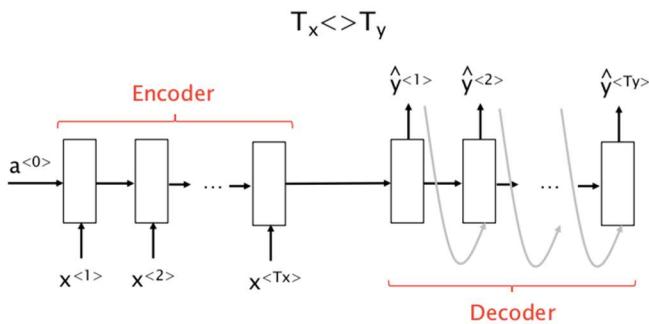
Diverse tipi di applicazione, caratterizzate da lunghezze di input e di output.



Nel caso della machine translation, ad esempio, la dimensione di input può essere diversa da quella di output;  
-> dall'input non posso produrre direttamente l'output.

Tipicamente si passano le attivazioni ad un'altra rete che si occupa di produrre l'output.

Si ha quindi in input un encoding e l'output fa un decoding.



**Encoder:** prima componente, si occupa di leggere la frase.  
(architettura standard many-to-many con parte di output tagliata)  
- processa tutto l'input e lo trasforma in uno stato nascosto che passa al decoder

**Decoder:** seconda componente, si occupa di fare una decodifica. (architettura one to many in quanto prende in input l'output del precedente)

### Many-to-many

E.g., machine translation

Nota: input e output non devono essere perforza dello stesso tipo. Posso avere in input immagini, in output testo, etc...  
L'architettura encoder-decoder è fondamentale per modelizzare il linguaggio.

### Language modeling

Modelli che siano anche in grado di generare un testo, capendo quali combinazioni di parole sono plausibili e quali no. Si basano su una probabilità che una serie di parole concorrono in una stessa frase.

The apple and pair salad

The apple and pear salad

$P(\text{The apple and pair salad}) = 3.2 \times 10^{-13}$

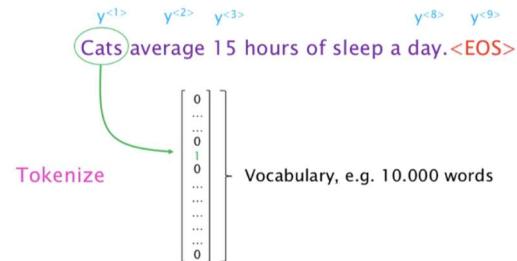
$P(\text{The apple and pear salad}) = 5.7 \times 10^{-10}$  Ad ogni frase viene associata una probabilità -> quali combinazioni di parole sono più probabili di altre.

$$P(\text{sentence}) = P(y^{<1>} | y^{<2>} | \dots | y^{<Ty>})$$

### Language modeling con RNN

Si parte da un training-set che è un insieme di frasi (es: Wikipedia)

Training set: large corpus of, e.g., English text



Questo insieme(corpous) è caratterizzato da un insieme di frasi, ciascuna frase sarà caratterizzata da un insieme di token e da un elemento che chiude la frase (es: il punto.).

La prima cosa che si fa è andare a tokenizzare (trasformare in insiemi numerici) la frase (spezzettare) sottoforma di one-hot vectors. Aggiungiamo, infine, un token chiamato <EOS> per indicare la fine della sequenza.

- Nel caso di parole che non conosciamo, possiamo usare dei token speciali per modelizzare parole che sono fuori dal nostro vocabolario.

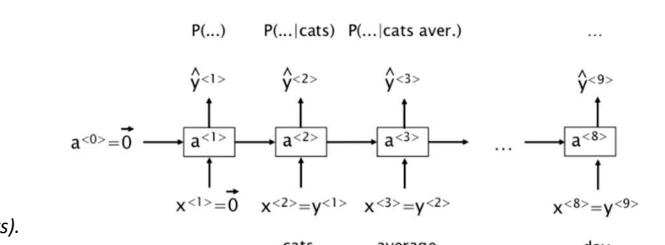
### Uso la RNN per modellare la probabilità.

Questo sarà il language model:

- Parte con hidden state nullo e con input nullo

Ci sono delle parole che possono seguire la parola Cats, altri meno frequentemente associati:

- Stiamo stimando una probabilità condizionata (es: *stimo probabilità che ci sia la parola "average" dopo aver visto la parola cats*). Questo meccanismo lo si ripete per la terza e per tutte le parole successive fino al token EOS.



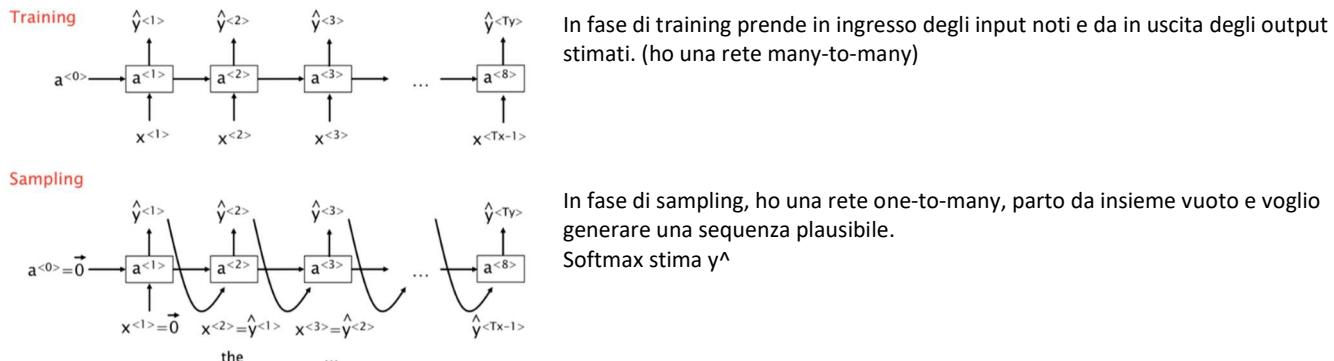
$$\text{E.g., } P(y^{<1>} | y^{<2>} | y^{<3>}) = P(y^{<1>}) P(y^{<2>} | y^{<1>}) P(y^{<3>} | y^{<1>} | y^{<2>})$$

**Nota:** come funzione di Loss si può usare una cross-entropy loss tra le possibili parole e quella che osserviamo nel nostro testo

Cats average 15 hours of sleep a day. <EOS>

Bisogna capire come usare questo modello per generare, campionare, nuove sequenze.

### Sentence generation (sampling)



Nel sampling vado a campionare nella distribuzione di probabilità prodotte da un determinato step passando la parola al layer successivo, il quale produrrà la parola seguente, etc. Si parla di sampling perché questo da la variabilità di generare sempre qualcosa di diverso. (infatti non si prende sempre la parola a probabilità maggiore altrimenti si genererebbe sempre la stessa frase)

### Character-level language model

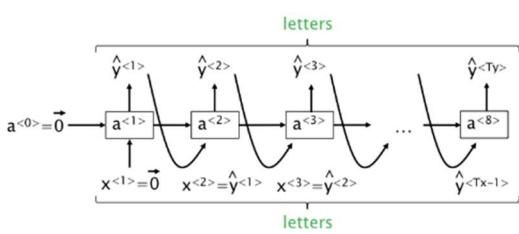
Lo stesso che si è fatto con le parole lo si può fare a livello di caratteri.

Vantaggio: se non ho tante parole

Svantaggio: devo memorizzare anche ortografie parole, sequenze molto lunghe

Word vocabulary: [a,aaron,...,zulu,<UNK>]

Character vocabulary: [a,b,...,z, ,;,...,0,...,9,A,...,Z]



tyntd-iafhatawiaoahrdemot lytdws e .tfti, astai f ogoh eoase rrranbyne 'nhthnee e  
plia tkldrgd t o idoe ns,smtt h ne etie h,hregtrs nigtike,aoaenns lng

↓ Train more

"Tmont thithey" fomesscerliund  
Keushey. Thom here  
sheulke, ammerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome  
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."

↓ Train more

Aftair fall unsuch that the hall for Prince Velzonski's that me of  
her heary, and behs to so arwage fiving were to it beloge, pavu say falling misfort  
how, and Gogition is so overelical and ofter.

↓ Train more

"Why do what that day," replied Natasha, and wishing to himself the fact the  
princess, Princess Mary was easier, fed in had oftened him.  
Pierre aking his soul came to the packs and drove up his father-in-law women.

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.

VIOLA:  
Why, Salisbury must find his flesh and thought  
That which I am not aps, not a man and in fire,  
To show the reining of the raven and the wars  
To grace my hand reproach within, and not a fair are hand,  
That Caesar and my goodly father's world;  
When I was heaven of presence and our fleets,  
We spare with hours, but cut thy council I am great,  
Murdered and by the master's ready there  
My power to give thee but so much as hell:  
Some service in the noble bondman here,  
Would show him to her wine.

KING LEAR:  
O, if you were a feeble sight, the courtesy of your law,  
Your sight and several breath, will wear the gods  
With his heads, and my hands are wonder'd at the deeds,  
So drop upon your lordship's head, and your opinion  
Shall be against your honour.

For  $\bigoplus_{i=1,\dots,n} \mathcal{L}_{m_i}$  where  $\mathcal{L}_{m_i} = 0$ , hence we can find a closed subset  $\mathcal{H}$  in  $\mathcal{H}$  and any sets  $\mathcal{F}$  on  $X$ ,  $U$  is a closed immersion of  $S$ , then  $U \rightarrow T$  is a separated algebraic space.

*Proof.* Proof of (1). It also start we get

$$S = \text{Spec}(R) = U \times_X U \times_X U$$

and the comparisonly in the fibre product covering we have to prove the lemma generated by  $\coprod Z \times_U U \rightarrow V$ . Consider the maps  $M$  along the set of points  $Sch_{fpf}$  and  $U \rightarrow U$  is the fibre category of  $S$  in  $U$  in Section ?? and the fact that any  $U$  affine, see Morphisms, Lemma ???. Hence we obtain a scheme  $S$  and any open subset  $W \subset U$  in  $Sch(G)$  such that  $\text{Spec}(R') \rightarrow S$  is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that  $f_i$  is of finite presentation over  $S$ . We claim that  $\mathcal{O}_{X,S'}$  is a scheme where  $x, x', s'' \in S'$  such that  $\mathcal{O}_{X,S'}$  is separated. By Algebra, Lemma ?? we can define a map of complexes  $\text{GL}_G(x'/S'')$  and we win.  $\square$

To prove study we see that  $\mathcal{F}|_U$  is a covering of  $X'$ , and  $\mathcal{T}_i$  is an object of  $\mathcal{F}|_{X/S}$  for  $i > 0$  and  $\mathcal{F}_p$  exists and let  $\mathcal{F}_i$  be a presheaf of  $\mathcal{O}_X$ -modules on  $\mathcal{C}$  as a  $\mathcal{F}$ -module. In particular  $\mathcal{F} = U/\mathcal{F}$  we have to show that

$$\tilde{\mathcal{M}}^* = \mathcal{I}^* \otimes_{\text{Spec}(k)} \mathcal{O}_{S,S} - i_X^{-1} \mathcal{F}$$

is a unique morphism of algebraic stacks. Note that

$$\text{Arrows} = (\text{Sch}/S)_{fpf}, (\text{Sch}/S)_{fpf}$$

and

$$V = \Gamma(S, \mathcal{O}) \rightarrow (U, \text{Spec}(A))$$

is an open subset of  $X$ . Thus  $U$  is affine. This is a continuous map of  $X$  is the inverse, the groupoid scheme  $S$ .

*Proof.* See discussion of sheaves of sets.  $\square$

The result for prove any open covering follows from the less of Example ???. It may replace  $S$  by  $X_{spaces,etale}$  which gives an open subspace of  $X$  and  $T$  equal to  $S_{Zar}$ , see Descent, Lemma ???. Namely, by Lemma ?? we see that  $R$  is geometrically regular over  $S$ .

**Lemma 0.1.** Assume (3) and (3) by the construction in the description. Suppose  $X = \lim |X|$  (by the formal open covering  $X$  and a single map  $\text{Proj}_X(A) = \text{Spec}(B)$  over  $U$  compatible with the complex

$$S\ell(A) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X})$$

When in this case of to show that  $Q \rightarrow C_{2/X}$  is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If  $T$  is surjective we may assume that  $T$  is connected with residue fields of  $S$ . Moreover there exists a closed subspace  $Z \subset X$  of  $X$  where  $U$  in  $X'$  is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1)  $f$  is locally of finite type. Since  $S = \text{Spec}(R)$  and  $Y = \text{Spec}(R)$ .

*Proof.* This is form all sheaves of sheaves on  $X$ . But given a scheme  $U$  and a surjective étale morphism  $U \rightarrow X$ . Let  $U \cap U = \coprod_{i=1,\dots,n} U_i$  be the scheme  $X$  over  $S$  at the schemes  $X_i \rightarrow X$  and  $U = \lim_i X_i$ .  $\square$

The following lemma surjective restcomposes of this implies that  $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{X,\dots,B}$

**Lemma 0.2.** Let  $X$  be a locally Noetherian scheme over  $S$ ,  $E = \mathcal{F}_{X/S}$ . Set  $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$ . Since  $\mathcal{I}^* \subset \mathcal{I}^*$  are nonzero over  $i_0 \leq p$  is a subset of  $\mathcal{J}_{n,0} \circ \tilde{A}_2$  works.

**Lemma 0.3.** In Situation ???. Hence we may assume  $q' = 0$ .

*Proof.* We will use the property we see that  $p$  is the next functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_X) = \mathcal{O}_X(D)$$

where  $K$  is an  $F$ -algebra where  $\delta_{n+1}$  is a scheme over  $S$ .  $\square$

```
static void do_command(struct seq_file *m, void *v)
{
    int column = 32 << (cmd[2] & 0x80);
    if (state)
        cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
    else
        seq = 1;
    for (i = 0; i < 16; i++) {
        if (k & (1 << 1))
            pipe = (in_use & UMXTHREAD_UNCCA) +
                ((count & 0x00000000ffffff8) & 0x0000000f) << 8;
        if (count == 0)
            sub(pid, ppc_md.kexec_handle, 0x20000000);
        pipe_set_bytes(i, 0);
    }
    /* Free our user pages pointer to place camera if all dash */
    subsystem_info = &of_changes[PAGE_SIZE];
    rek_controls(offset, idx, &soffset);
    /* Now we want to deliberately put it to device */
    control_check_polarity(&context, val, 0);
    for (i = 0; i < COUNTER; i++)
        seq_puts(s, "policy ");
}
```

Questo tipo di modelli riesce facilmente a generare sequenze plausibili ma può avere difficoltà a gestire una grammatica molto precisa, se noi lo usiamo per scrivere codice, ci potrebbero essere errori come parentesi mancanti, inconsistenze a livello logico...

Saltato ..

Visualizzare le RNN

```
/* 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.
    */


```

"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."

## Quote (" ") detection cell

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.

## Line length tracking cell

```
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;
}
```

## "if" statement cell

```
/* Duplicate LSM field information. The lsm_rule is opaque, so
 * re-initialized. */
static inline int audit_dupe_lsm_field(struct audit_field *df,
                                      struct audit_field *sf)
{
    int ret = 0;
    char *lsm_str;
    /* Our own copy of lsm_str */
    lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
    if (unlikely(!lsm_str))
        return -ENOMEM;
    df->lsm_str = lsm_str;
    /* Our own (refreshed) copy of lsm_rule */
    ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
                                   (void *)df->lsm_rule);
    /* Keep currently invalid fields around in case they
     * become valid after a policy reload. */
    if (ret == -EINVAL) {
        pr_warn("audit rule for LSM \\'%s\\' is invalid\n",
                df->lsm_str);
        ret = 0;
    }
}
return ret;
```

## Comment cell

```
#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;
}
```

## Code depth cell

# MODELLING DYNAMICAL SYSTEMS

Un sistema dinamico è un oggetto matematico usato per descrivere l'evoluzione nel tempo di variabili che caratterizzano dei fenomeni sotto studio.

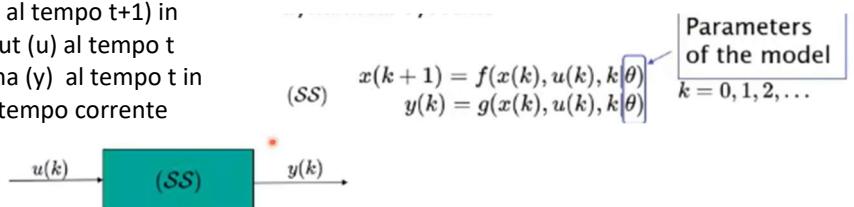
- Parliamo di sistemi dinamici nel tempo continuo -> modello descritto con equazioni differenziali ordinarie  $\frac{dx}{dt} = f(x(t), u(t), t) \quad t \in \mathbb{R}, t \geq 0$
- Parliamo di sistemi dinamici nel tempo discreto -> modello descritto con difference equations  $x(k+1) = f(x(k), u(k), k) \quad k = 0, 1, 2, \dots$

Hanno molte applicazioni, dove vogliamo fare delle predizioni del comportamento del sistema per poi disegnare dei sistemi di controllo -> indirizzare il comportamento del sistema verso un output specifico.

## State Space representation:

descrizione generale del sistema:

- Funzione per regolare  $x$  (stato di sistema al tempo  $t+1$ ) in funzione dello stato al tempo  $t$  e dell'input ( $u$ ) al tempo  $t$
- Funzione per regolare l'output ( $y$ ) al tempo  $t$  in funzione del suo stato e del suo input al tempo corrente



Ci si riconduce ad una rappresentazione che è una forma regressiva in cui l'uscita al tempo  $t$  dipende da tutti gli input e da tutte le uscite ai tempi precedenti, in sequenza.

- Comoda perché non richiede di modellizzare un vettore di stato
  - o Ci da una relazione diretta tra ingressi e uscite del sistema

$$(\mathcal{R}) \quad y(k) = \phi(y(k-1), \dots, y(k-n), u(k), \dots, u(k-n), k, \theta)$$

Parameters of the model

Dunque, ciò che vogliamo fare è **system identification**: vogliamo apprendere da dei dati ottenuti sperimentalmente, una modellizzazione del nostro sistema. Ad esempio, se abbiamo un sistema robotico, lo sollecitiamo con un input, registriamo l'uscita e usiamo questi dati -che sono affetti da rumore- per imparare un modello; possiamo usare una rete neurale per imparare a rifare questa simulazione, sfruttando la sua capacità di approssimazione.

Per fare questo usiamo l'errore di predizione, predo ciò che succede al tempo  $t+1$  e usiamo l'errore per ottimizzare la rete neurale -> 1-step ahead prediction error: predico ciò che succederà al passo successivo e poi misuro l'errore. (osservo fino a  $t-1$ )

$$MSE_{pred} = \frac{1}{N_{test}-k_0+1} \sum_{k=k_0}^{N_{test}} [y_{test}(k) - y_{model}(k|y_{test}, u_{test}, \theta)]^2$$

evaluates the output of the model at time  $k$  using past values of the **test** dataset.

**Nota:** in fase di simulazione, la situazione è diversa, ai passi successivi ci si basa sull'output si usa effettivamente l'uscita della rete -> nel momento in cui arrivo a predirre  $y(t)$ , gli errori si saranno accumulati -> l'errore che io compio sulla simulazione è maggiore rispetto all'errore che compio nella fase di train utilizzando solo 1-step ahead.

Infatti molto spesso, quello che si fa è utilizzare delle protocolli di training in cui la rete tende ad imparare sul proprio output in modo tale da correggere gli errori o perlomeno evitare di propagarli in maniera catastrofica.

Questo tipo di problemi potrebbe essere modelizzato con una rete neurale statica che prende in input gli input fino al tempo  $n$  e da in uscita la simulazione dal tempo  $n$  al tempo  $N$  ma questo tipo di rete neurale non è particolarmente utile (non ha forma di memoria, non si adatta facilmente a simulazioni di lunghezza variabile... conviene usare le reti neurali ricorrenti).

## Rete neurale ricorrente viene scelta in quanto:

- Prende in ingresso l'ingresso al tempo  $t$
- è in grado di utilizzare le proprie predizioni fino al tempo  $t-1$  per completare la simulazione

|              | state-space model | RNN model                   |
|--------------|-------------------|-----------------------------|
| inputs       | $u(k)$            | $x(k)$                      |
| state vector | $x(k)$            | $h_1(k), \dots, h_{L_R}(k)$ |
| outputs      | $y(k)$            | $y(k)$                      |

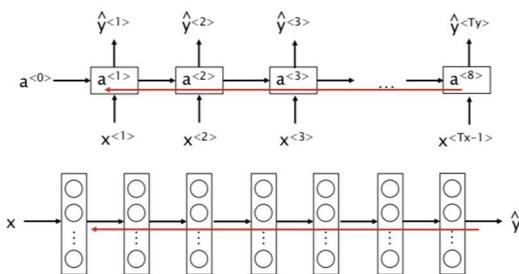
addestrare le reti neurali può essere difficile a causa del vanishing gradient

### Vanishing gradients con le RNN

Reti molto lunghe rischiano di soffrire di vanishing gradients o anche di exploding gradients e si possono perdere informazioni importanti.

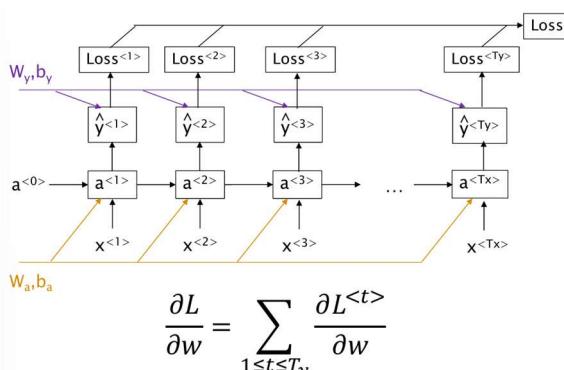
The **cat**, which already ate ..., **was** full.

The **cats**, which already ate ..., **were** full.



Il verbo è legato ad una parola molto vecchia, ho bisogno che questa informazione di propaghi fino al verbo -> ho bisogno che il gradiente non svanisca.

La back propagation deve trasferire questa informazione indietro nella rete (gradiente potrebbe esplodere o imploredere). Posso decidere che se gradiente sopra certa soglia lo fermo, con i vanishing gradient è complicato perché smetto di imparare.



Se abbiamo  $t$  istanti temporali, i nostri gradienti sono la somma di tutti i componenti delle Loss, dobbiamo vedere cosa succede su ciascuno di questi componenti

$$\frac{\partial L}{\partial w} = \sum_{1 \leq t \leq T_y} \frac{\partial L^{<t>}}{\partial w}$$

Possiamo esprimere la loss come il prodotto di  $n$  componenti:

$$\frac{\partial L^{<t>}}{\partial w} = \sum_{1 \leq k \leq t} \frac{\partial L^{<t>}}{\partial a^{<k>}} \frac{\partial a^{<t>}}{\partial a^{<k>}} \frac{\partial a^{<k>}}{\partial w}$$

Posso esprimere i gradienti come un prodotto delle attivazioni per tutti i tempi precedenti:

$$\frac{\partial a^{<t>}}{\partial a^{<k>}} = \prod_{i \leq k \leq t} \frac{\partial a^{<i>}}{\partial a^{<i-1>}} = \prod_{i \leq k \leq t} W_{aa}^T \sigma'(a^{<i-1>})$$

Dunque, per calcolare la Loss, sto moltiplicando il vettore di pesi per se stesso tante volte quanti sono gli step della nostra sequenza.

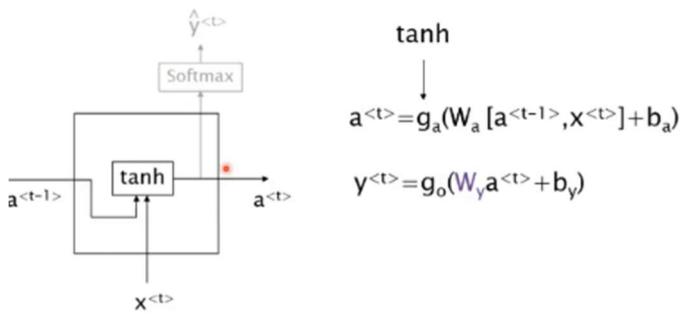
Waa "small" → Vanishing gradients

Waa "large" → Exploding gradients

Questo tipo di architetture ha quindi questo limite strutturale che si può parzialmente correggere facendo del fine-tuning ma non si può completamente eliminare -> ci sono altre architetture utilizzabili.

$$\frac{\partial a^{<t>}}{\partial a^{<k>}} = \prod_{i \leq k \leq t} \frac{\partial a^{<i>}}{\partial a^{<i-1>}} = \prod_{i \leq k \leq t} W_{aa}^T \sigma'(a^{<i-1>})$$

## RNN unit



**Wa**: matrice di pesi che permette di regolare l'hidden state in funzione dell'hidden-state al tempo t-1 e dell'input al tempo t.  
**g<sub>a</sub>**: funzione di attivazione non lineare (es: tanh)

altra funzione: prende in ingresso gli hidden-state al tempo t e ci da in uscita y utilizzando una funzione di attivazione g<sub>o</sub> che nel caso di problemi di classificazione è una softmax.

Modifichiamo questa unità base usando GRU che soffre meno il vanishing gradient.

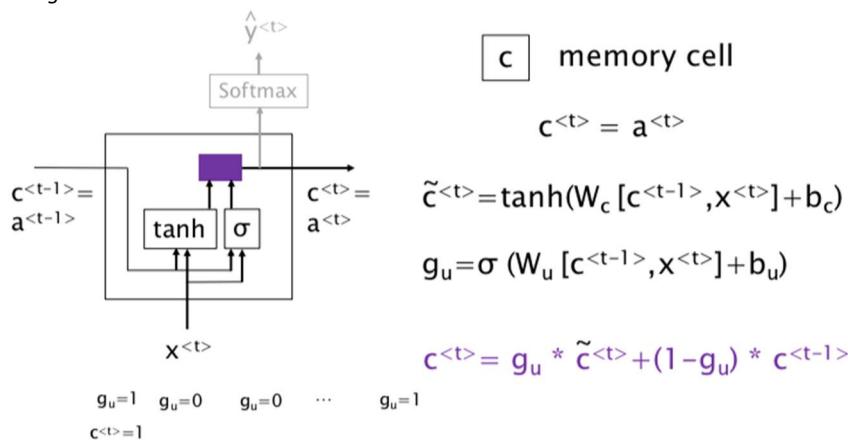
## GRU (Gated Recurrent Unit) semplificata

La GRU introduce un concetto di **memory cell**: ad ogni time step calcolo  $\tilde{c}$  (l'attivazione dell'unità RNN) e in più vado a decidere se aggiornare o meno l'informazione con  $g_u$  calcolato applicando una funzione non lineare.

Immaginiamo di avere una memory cell che dice alla rete se debba ricordare qualcosa della parola.

Inizialmente la rete potrebbe assegnare un valore 1 a questa memory cell e usa  $g_u$  per decidere se aggiornare o meno la memory cell.

È come se la rete impara a non cambiare una certa cella fino a quando non ne trova un'altra da memorizzare e quindi capisce che le due sono collegate.



**c** memory cell

-> ad ogni timestamp valuto se sovrascrivere o meno il valore della memory cell andando a calcolare  $\tilde{c}$ .

->  $g_u$  da un'indicazione su quanto è conveniente aggiornare la memoria (viene calcolato dalla rete in funzione del contenuto attuale della cella e in funzione dell'input corrente) (0 dimentica tutti, 1 ricorda, intermedi bho)

The cat, which already ate ..., was full.

-> quando arriviamo a was/were dobbiamo tenere in considerazione quello che abbiamo visto agli istanti precedenti per processare l'input corrente e dare un'uscita -> lo facciamo attraverso una somma persata tra il valore di  $c^{<t-1>}$  e  $\tilde{c}$  (come peso usiamo il valore di  $g_u$ ). Questo ci permettere di sopprimere tutto ciò che c'è all'interno dell'inciso per ricordarci di "the cat".  
- sarà la rete ad imparare i valore di gate, non sarò io ad impostarli in maniera forzata.

A questo punto abbiamo la nostra uscita che andiamo a calcolare a partire dal contenuto della cella di memoria al tempo t.

## GRU full

$g_r$ : relevance gate -> mi dice quanto il valore che vado ad utilizzare dal passato (cella di memoria corrente) sia rilevante.  
Tutto ciò significa aggiungere dei parametri che la rete dovrà imparare.

$$\tilde{c}^{<t>} = \tanh(W_c [g_r * c^{<t-1>} , x^{<t>}] + b_c)$$

$$g_u = \sigma (W_u [c^{<t-1>} , x^{<t>}] + b_u)$$

$$g_r = \sigma (W_r [c^{<t-1>} , x^{<t>}] + b_r)$$

$$c^{<t>} = g_u * \tilde{c}^{<t>} + (1-g_u) * c^{<t-1>}$$

Dunque, i **gate** sono dei fattori di modulazione che ci permettono di allungare, quando serve, la lunghezza effettiva della sequenza.

## GRU e LSTM (Long Short-Term Memory)

LSTM è parente di GRU, un po' più sofisticato ma non si usa il  $g_r$ , c'è comunque **gu applicato per le attivazioni**. Ci sono altri **due gate di forget  $g_f$**  e di **output  $g_o$**  e l'uscita  $c$  viene ottenuta prendendo anche il  $g_f$  perché posso sfruttare insieme quello calcolato al livello attuale più quello del passato.

GRU

$$\tilde{c}^{<t>} = \tanh(W_c [g_r * c^{<t-1>}, x^{<t>}] + b_c)$$

$$g_u = \sigma (W_u [c^{<t-1>}, x^{<t>}] + b_u)$$

$$g_r = \sigma (W_r [c^{<t-1>}, x^{<t>}] + b_r)$$

$$c^{<t>} = g_u * \tilde{c}^{<t>} + (1 - g_u) * c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$

LSTM

$$\tilde{c}^{<t>} = \tanh(W_c [a^{<t-1>}, x^{<t>}] + b_c)$$

→ Candidato

$$g_u = \sigma (W_u [a^{<t-1>}, x^{<t>}] + b_u)$$

→ Gate di update

$$g_f = \sigma (W_f [a^{<t-1>}, x^{<t>}] + b_f)$$

→ Gate di forget

$$g_o = \sigma (W_o [a^{<t-1>}, x^{<t>}] + b_o)$$

→ Gate di output

$$c^{<t>} = g_u * \tilde{c}^{<t>} + g_f * c^{<t-1>}$$

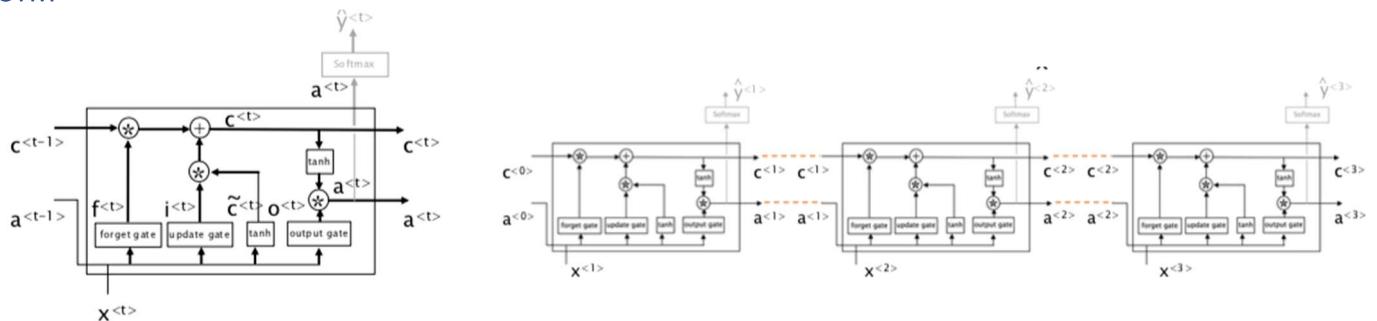
→ Cella al tempo corrente

→

$$a^{<t>} = g_o * \tanh(c^{<t>})$$

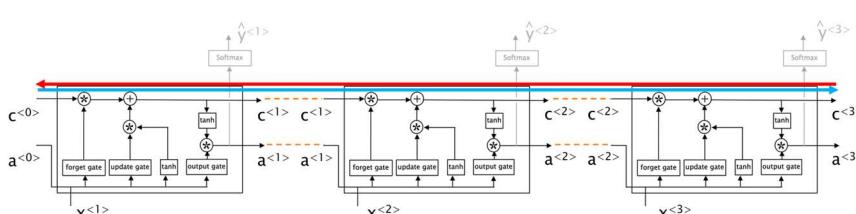
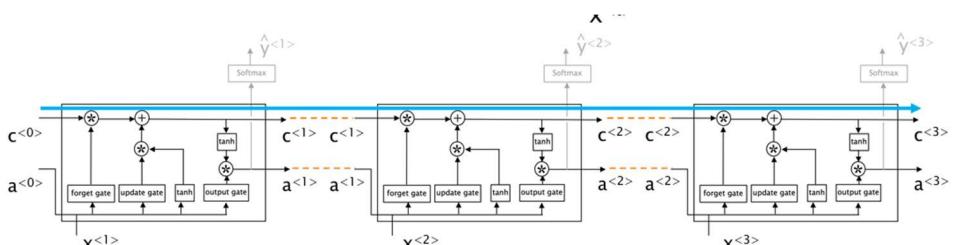
NOTA: con LSTM **non c'è alcun vincolo** che  $g_u+g_f$  faccia 1, quindi posso ricordare/dimenticare quel che voglio.

LSTM



Nota: collegandola una dietro l'altra, ci sono due flussi di informazione:

- **Linea azzurra:** passaggio in alto che regola come l'informazione viene propagata nella cella di memoria
- **Linea tratteggiata in basso:** regola il modo in cui l'informazione viene passata attraverso l'hidden-state.



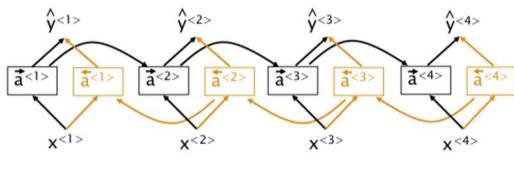
Anche la **backward propagation** passa su entrambi i percorsi, sul percorso rosso ho una moltiplicazione per un gate che può avere dei valori diversi per ogni timestamp (a differenza della RNN standard dove la moltiplicazione avveniva per un vettore di pesi sempre uguale e quindi tendeva ad esplodere/implodere) ~ meccanismo di memoria

Dunque, queste reti risolvono anche il problema della vanishing gradients.

È come se ci fosse una fast forward che porta avanti informazioni nella rete ma al tempo stesso si ha col  $g_f$  una variabilità che blocca l'esplosione o il vanishing del gradiente.

## Bidirectional RNN

Per fare predizioni sia in avanti che all'indietro si aggiunge un time step per la direzione opposta dove ricevono anch'essi gli input. (quello che vediamo non è un backward propagation ma un forward propagation -> Il forward viene fatto anche da destra verso sinistra.). quando faremo bacward propagation, avremo dunque 2 flussi, uno in indietro e uno in avanti.



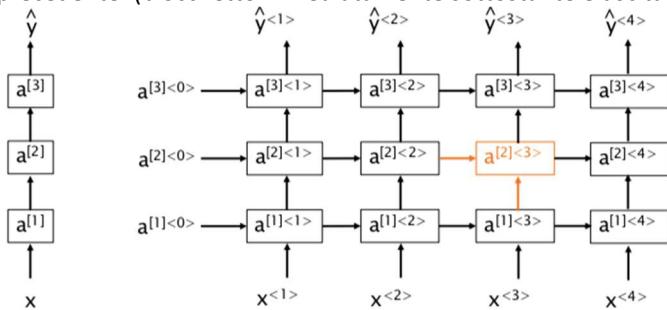
$$\hat{y}^{<t>} = g(W_y[\hat{a}^{<t>}, \hat{a}^{<t>}], b_y)$$

Lo svantaggio è la latenza perché bisogna aspettare la fine della frase per fare la predizione.

Mettendo delle LSTM una sopra l'altra, posso creare delle Deep RNN

## Deep RNN

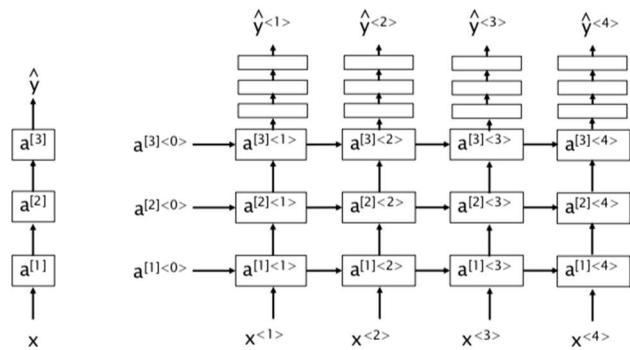
L'ingresso al tempo t, viene dato in ingresso a tutti i layer della RNN che prendono anche in ingresso le attivazione del layer precedente. (blocchetto immediatamente sottostante e uscita di quello alla sua sx)



$$a^{[2]<3>} = g(W_a^{[2]} [a^{[2]<2>}, a^{[1]<3>}] + b_a^{[2]})$$

Molto onerosa, la complessità di questi oggetti esplode, solitamente si trovano massimo 2/3 livelli quindi 2/3 LSTM una sopra l'altra.

Se ci fosse bisogno di ulteriore complessità, si potrebbero aggiungere dei layer fully connected che dipenderanno totalmente dall'uscita al tempo corrente e non avrà connessioni laterali (per raffinare ulteriormente l'output della rete)-



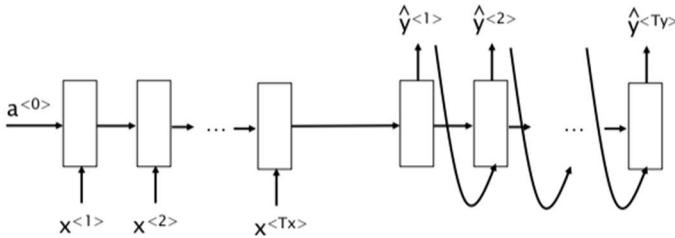
## Attention model

Si applicano al modello encoder decoder e il problema delle reti viste fin'ora è che bisogna dare in pasto tutta la frase per avere il risultato.

Jane s'est rendue en Afrique en septembre dernier, a apprécié la culture et a rencontré beaucoup de gens merveilleux; elle est revenue en parlant comment son voyage était merveilleux, et elle me tente d'y aller aussi.

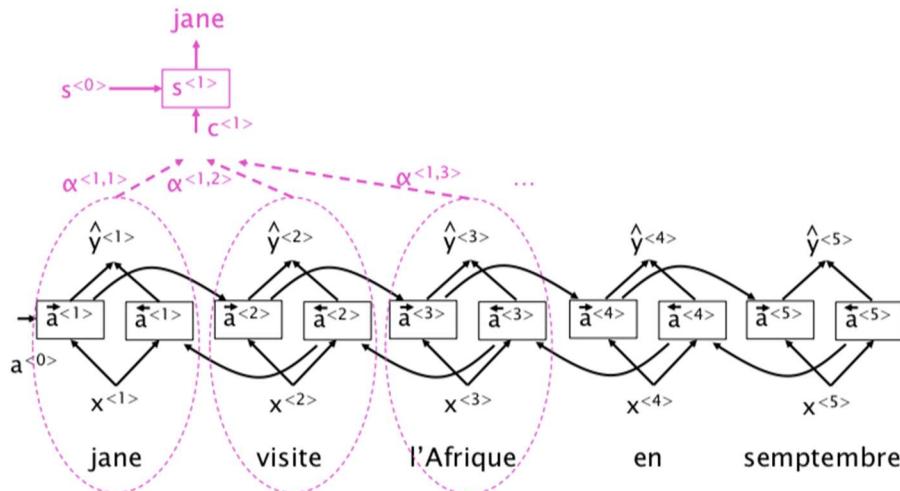
Jane went to Africa last September, and enjoyed the culture and met many wonderful people; she came back raving about how wonderful her trip was, and is tempting me to go too.

Voglio tradurre da francese ad inglese.



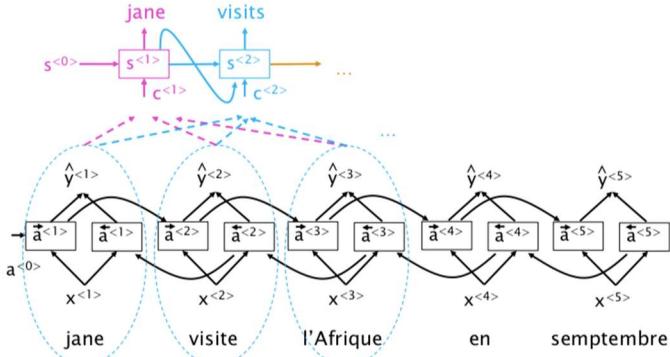
Si buttano via gli output ed è come se ci fosse un'altra rete che prende in input le informazioni dei singoli time steps e dice: "quando la mia rete sta cercando di produrre l'output per la parola quali delle parole sta guardando?".

Inserisco dei **pesi  $\alpha$  (coefficienti di attenzione)** che considerano il contributo proveniente dalle attivazioni dei livelli precedenti e descrivono quello che negli attention model è il contesto. -> vengono presi al momento corrente per capire qual è la predizione più accurata (i dettagli a cui fare attenzione dipendono dal contesto).



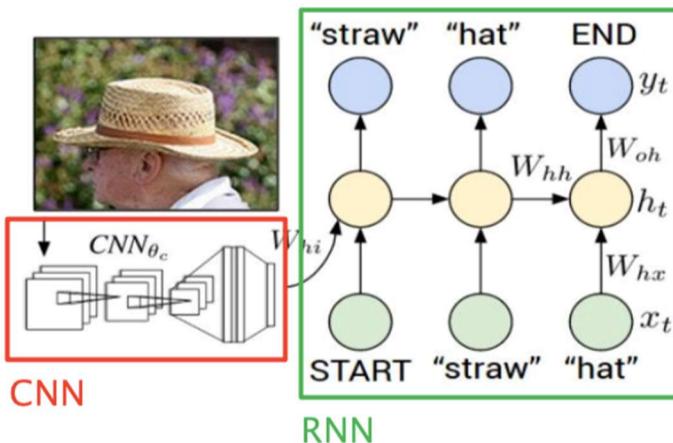
Introduce un grado di libertà in più che permette di spostare il focus su ciò che è rilevante e sopprimere ciò che non lo è.

In questo modo la rete concentra la propria attenzione solo su alcune parti.

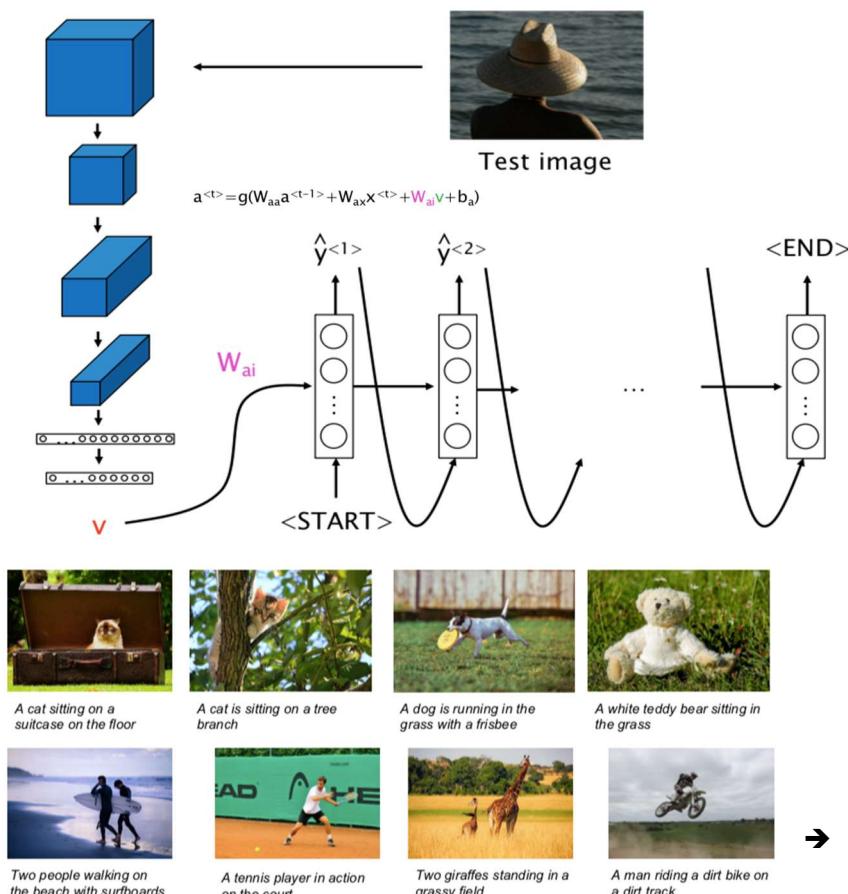


Questo tipo di meccanismo è stato proposto ed è utile anche su sistemi misti. Ad esempio, sistemi che prendono in ingresso un'immagine e danno in uscita una descrizione.

## RNN e image captioning



CNN genera un embedding dell'immagine che dà alla RNN che genera una caption dell'immagine, una parola alla volta.



A test time andiamo a prendere la nostra immagine e poi utilizziamo la RNN per generare una caption come abbiamo visto prima

- Parto da token di stati che mi dice di iniziare a generare la caption
- All'hidden state do in ingresso anche l'embedding dell'immagine
- Così via

## Success cases ...



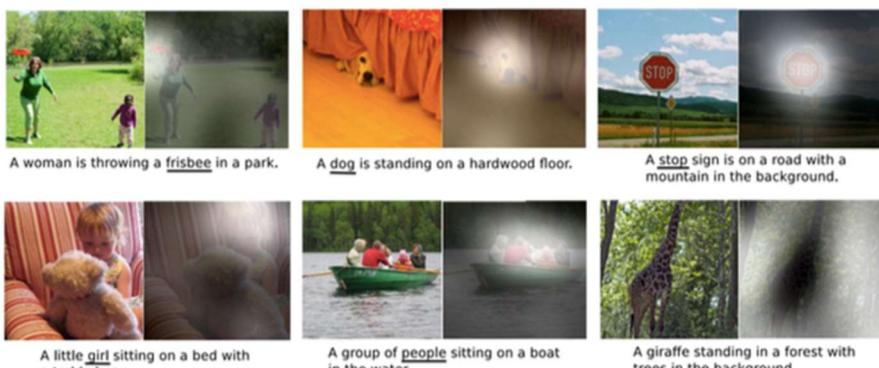
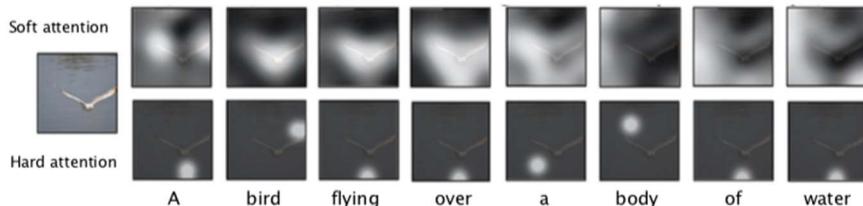
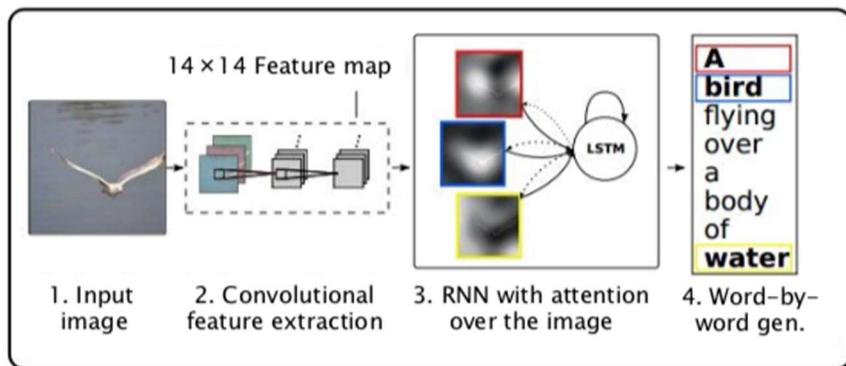
→ Cose più comuni

→ Cose meno comuni

## ... and failure cases

## Image captioning with attention

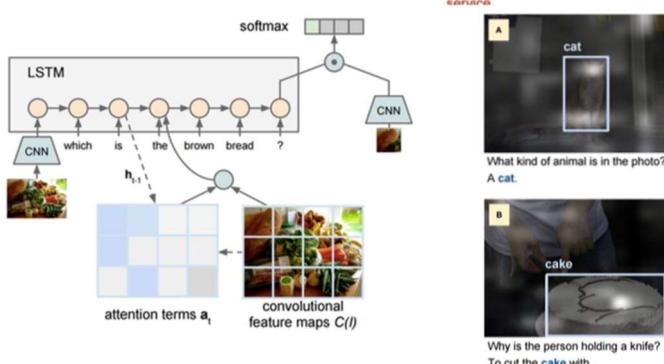
L'attenzione è utile in questo tipo di modelli perché ci sono parole che si riferiscono ad una parte specifica dell'immagine. Ci aiuta a sopprimere alcune parti e a focalizzarsi su altre.



→ Ad ogni parola corrisponde un Attention map che può essere visualizzato  
-> questo ci aiuta a capire se il modello ha associato la parola alla parte corretta dell'immagine.

Soft attention: livelli tra 0 e 1 (non discreto)

## Visual question answering: altri task -> combinazione di testo e immagini



## Transformer model: Attention is all you need

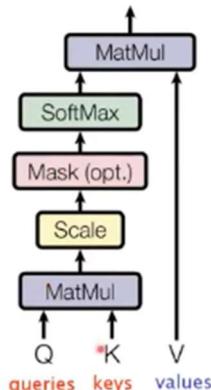
Transformer: architettura molto recente che porta il meccanismo dell'attenzione al più alto livello possibile.

- Butto via ricorrenza
- Uso l'attenzione per aprire quali parti della sequenza hanno delle affinità, tutta l'informazione viene codificata e processata attraverso dei moduli/blocchi base che si chiamano attention layer.

- Transformers are an alternative to recurrent neural networks that use only attention

- The building block is the scale dot-product attention

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK}{\sqrt{d_k}}\right)V$$



Trasformers sono costituiti da una sequenza in cui diversi blocchi di trasformazione vengono collegati uno dopo l'altro, a ciascun blocco viene fatta un'elaborazione, vengono presi i valori dello step precedente, vengono codificati con V e vengono moltiplicati con dei pesi che rappresentano i coefficienti di attenzione.

Nota: in base alla rilevanza di Queries e Keys bisogna capire a cosa fare attenzione.

- Idealmente devo pensare di avere 2 sequenze di token
  - Da 1 prendo i valori
  - Da 1 prendo le keys

FINO A MINUTI 1:59:10 secondo  
me non ha capito manco lei che sta dicendo.

Note:

- Queries and Keys sono vettori di dimensione dk
  - Queries: tutte le parole della frase che voglio tradurre
  - Key: parola da tradurre
    - ➔ Vado a capire quali parti e quali queries sono compatibili con la mia parola che voglio tradurre (per misurare compatibilità, misuro embedding delle keys e delle queries).
    - + compatibili -> amplificati
    - - compatibili -> soppressi

Nota: il confronto viene effettuato facendo un prodotto delle Queries e delle Keys

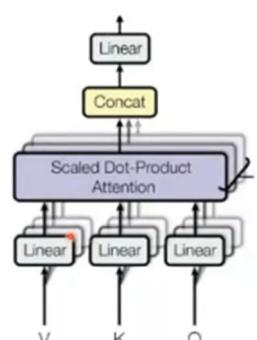
- Anche Values è un vettore di dimensione dk
- Dk è la dimensione degli embedding

Nota:

- Softmax utile perché ci assicura che la somma dei coefficienti di attenzione abbia somma 1
  - Devo focalizzarmi su qualcosa ma sopprimere qualcos'altro
- Matrici di riscalamento/pesi: effettuano un riscalamento dei valori di Queries e dei valori di Keys

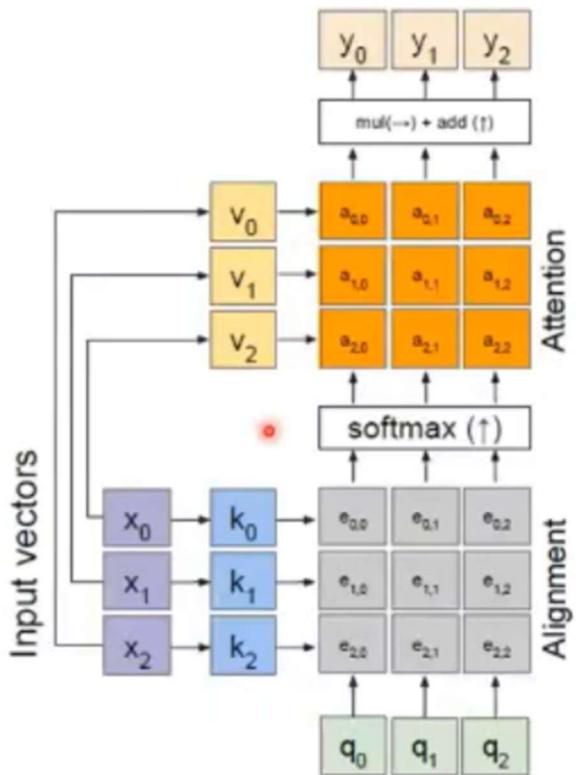
Nei trasformers considero una batteria di moduli di attenzione -> MULTI-HEAD ATTENTION:

- si parla di circa 8 head
- Ci consente di computare più contesti
  - Fare attenzione a più cose contemporaneamente

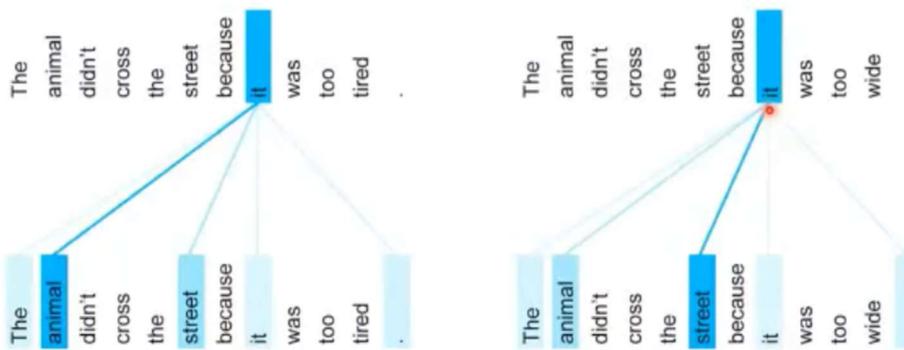


Nel modello di linguaggio, si fa l'assunzione che i vettori V, K, Q corrispondano alla stessa sequenza di token per cui il singolo elemento funziona sia da valore, sia da K sia da Q e consideriamo tutte le possibili combinazioni. (quantità di memoria è quadratica a quella della sequenza -> se ho una sequenza di 1000 token, per computare la matrice di attenzione dovrò computare una matrice 1000x1000 in cui vado a considerare la combinazione di ciascun elemento con tutti gli altri elementi della sequenza.)

Sulla base del contenuto della frase, per ciascuna parola della frase, decido quali componenti della frase sono rilevanti e quali no per la sua interpretazione -> SELF ATTENTION.



- Faccio questa computazione
- Applico la softmax per ottenere dei pesi tra 0 e 1
- Uso i pesi per trasformare i vettori di embedding
  - o Dunque ogni volta che applico questo layer di trasformer trasformo i vettori di embedding



Nota: nella nostra mente, è automatico capire che it fa parte dell'animale nella prima strada mentre alla strada nella seconda. L'attenzione ci consente di trasformare il vettore di embedding della parola it, pesando

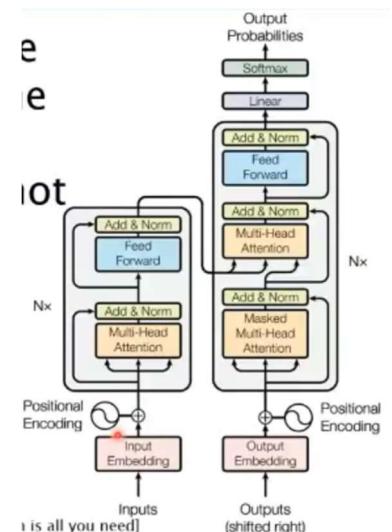
opportunamente il contesto; quindi, tenendo conto che nel primo caso it avrà una forte affinità con la parola animal, mentre nel secondo caso una forte affinità con la parola street. In entrambi i casi la matrice di attenzione non sopprime del tutto l'alternativa perché c'è una possibile ambiguità ma a seconda del contesto della frase è più probabile che sia associata ad una dei due casi.

L'architettura dei transfromer:

- Encoder
  - o Sequenza di blocchi di attenzione (2/3) che vengono applicati su un embedding dell'input
- Decoder

Uno dei problemi è che l'attention layer non dipende dall'ordine della sequenza:

- Matrice di attenzione viene calcolata moltiplicando gli embedding di tutte le parole per gli embedding di tutte le altre parole -> costruendo tutte le combinazioni di coppie -> l'ordine viene perso (a differenza della RNN che processa uno alla volta e quindi ha coscienza dell'ordine)
- Se vogliamo tenere conto dell'ordine, dobbiamo dare questa informazione alla rete
  - o Dobbiamo dare degli encoding posizionali -> posizione del token all'interno della sequenza
    - Codificati in maniera molto sofisticata



RNN:

- Ha problemi di vanishing gradient (va in difficoltà con lunghe sequenze)
- Aspetta sequenze ordinate di input
- Impiega un modello computazionale sequenziale (training time lungo)

Transformers:

- Buoni con lunghe sequenze e dipendenze a lungo raggio
- Possono operare su insieme non ordinati o con dati mancanti
- Impiegano un modello di computazione parallelo: tutte le computazioni possono essere fatte in parallelo
- 😅 richiedono tanta memoria, dati di training e risorse computazionali

Nota: i transformer possono essere usati anche per lavorare su immagini perché l'immagine può esser vista come una sequenza di parole.

- Immagine divisa in blocchettini 16x16 messi uno dietro l'altro e dati in pasto ai transformer
  - o Tratto immagine e testo come se fossero lo stesso tipo di dato come se fossero dei dati embedded
  - o Posso usare parti distanti tra immagini per comprenderle entrambe

