

IIC3670 Procesamiento de Lenguaje Natural

<https://github.com/marcelomendoza/IIC3670>

- POS, NER Y SEQUENCE LABELING -

Conceptos de NLP

POS tagging

- ▶ Etiquetar cada término de acuerdo a la función que este cumple en el texto.
- ▶ Puede ayudarnos en tareas como detección de estilo, parsing, detección de colocaciones.
- ▶ Tarea importante en NLP.

Text:

John likes the blue house at the end of the street .

Adjective	Determiner	Preposition
Adverb	Noun	Pronoun
Conjunction	Number	Verb

Conceptos de NLP

POS tagging

	Tag	Description	Example
Open Class	ADJ	Adjective: noun modifiers describing properties	<i>red, young, awesome</i>
	ADV	Adverb: verb modifiers of time, place, manner	<i>very, slowly, home, yesterday</i>
	NOUN	words for persons, places, things, etc.	<i>algorithm, cat, mango, beauty</i>
	VERB	words for actions and processes	<i>draw, provide, go</i>
	PROPN	Proper noun: name of a person, organization, place, etc..	<i>Regina, IBM, Colorado</i>
	INTJ	Interjection: exclamation, greeting, yes/no response, etc.	<i>oh, um, yes, hello</i>
Closed Class Words	ADP	Adposition (Preposition/Postposition): marks a noun's spacial, temporal, or other relation	<i>in, on, by under</i>
	AUX	Auxiliary: helping verb marking tense, aspect, mood, etc.,	<i>can, may, should, are</i>
	CCONJ	Coordinating Conjunction: joins two phrases/clauses	<i>and, or, but</i>
	DET	Determiner: marks noun phrase properties	<i>a, an, the, this</i>
	NUM	Numeral	<i>one, two, first, second</i>
	PART	Particle: a preposition-like form used together with a verb	<i>up, down, on, off, in, out, at, by</i>
	PRON	Pronoun: a shorthand for referring to an entity or event	<i>she, who, I, others</i>
	SCONJ	Subordinating Conjunction: joins a main clause with a subordinate clause such as a sentential complement	<i>that, which</i>
Other	PUNCT	Punctuation	<i>! , ()</i>
	SYM	Symbols like \$ or emoji	<i>\$, %</i>
	X	Other	<i>asdf, qwfg</i>

Conceptos de NLP

POS tagging en NLTK

```
>>> text = word_tokenize("And now for something completely different")
>>> nltk.pos_tag(text)
[('And', 'CC'), ('now', 'RB'), ('for', 'IN'), ('something', 'NN'),
 ('completely', 'RB'), ('different', 'JJ')]
```


```
>>> text = word_tokenize("They refuse to permit us to obtain the refuse permit")
>>> nltk.pos_tag(text)
[('They', 'PRP'), ('refuse', 'VBP'), ('to', 'TO'), ('permit', 'VB'), ('us', 'PRP'),
 ('to', 'TO'), ('obtain', 'VB'), ('the', 'DT'), ('refuse', 'NN'), ('permit', 'NN')]
```

Conceptos de NLP

POS tagging en Spacy (Español)

```
!python -m spacy download es_core_news_sm  
!python -m spacy download es_core_news_md
```

```
import spacy  
import es_core_news_sm  
import es_core_news_md  
  
nlp = es_core_news_sm.load()  
doc = nlp('''Un desastroso espíritu posee tu tierra:  
    donde la tribu unida blandió sus mazas,  
    hoy se enciende entre hermanos perpetua guerra,  
    se hieren y destrozan las mismas razas.''' )  
  
for token in doc: print(token.text, "|", token.lemma_, '|', token.pos_)
```



```
Un | Un | DET  
desastroso | desastroso | NOUN  
espíritu | espíritu | PROPN  
posee | poseer | VERB  
tu | tu | DET  
tierra | tierra | NOUN  
: | : | PUNCT
```

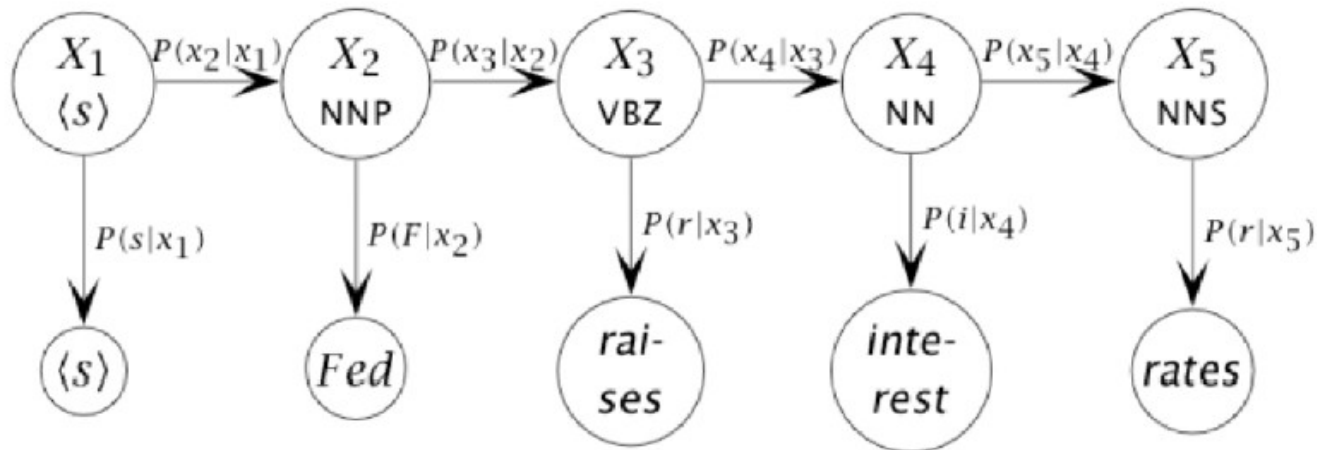
Conceptos de NLP

POS tagging con Hidden Markov Models (modelo clásico)

- ▶ Se dispone de un corpus etiquetado.
- ▶ La secuencia de tags es interpretada como una cadena de Markov:
 $P(x_{t+1} \mid x_t, \dots, x_1) = P(x_{t+1} \mid x_t)$, x_1, \dots, x_{t+1} representan tags
- ▶ Usamos un modelo generativo para términos, con tags como estados ocultos: $P(t \mid x_1, \dots, x_{t+1}) = P(t \mid x_{t+1})$

Conceptos de NLP

POS tagging con HMM (modelo clásico)



- En general muestran buena precisión (sobre 90 %).

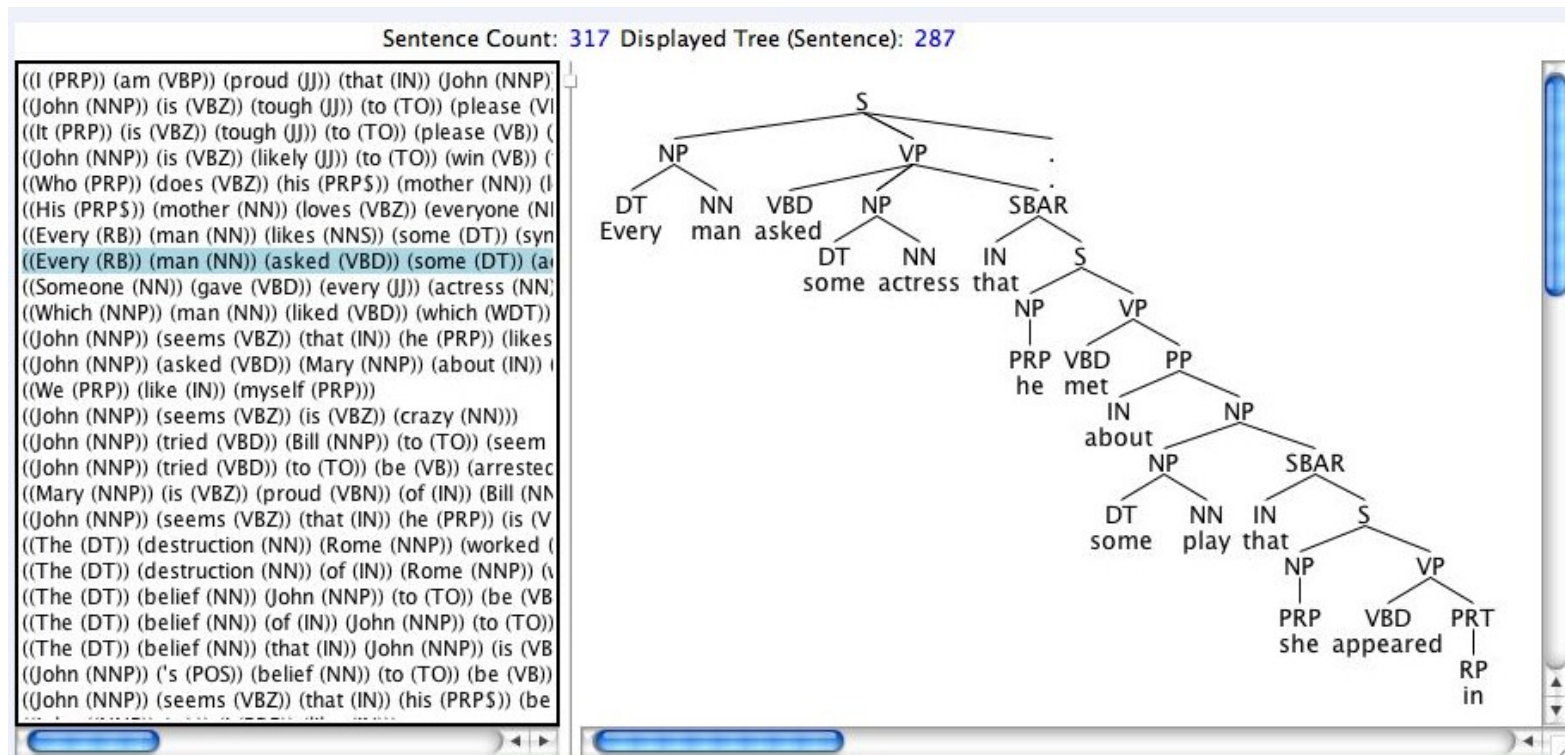
El algoritmo de entrenamiento se llama Viterbi.

Conceptos de NLP

POS tagging ¿Cuáles datos usan?

Treebanks: [Penn treebank](#) (más famoso), [UAM Spanish Treebank](#), ...

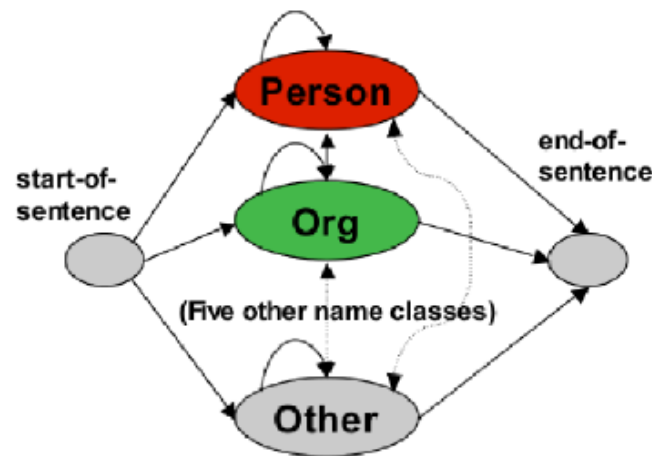
[Treebank viewer](#):



Conceptos de NLP

Named Entity Recognition (NER)

- ▶ Tarea: Identificar entidades en texto (personas, organizaciones, etc.)
- ▶ Separa el text en chunks, y para cada cual asocia una NE. Opera sobre texto tagged.
- ▶ NER types: organization, person, location, date, time, money, percent, facility (human made artifacts), gpe (geo-political ents).
- ▶ POS tagging puede ayudar, agregando *entity* como un estado mas.



Conceptos de NLP

Named Entity Recognition

bi-grama



1 President Xi Jinping of China, on his first state visit to the United States, showed off his familiarity with American history and pop culture on Tuesday night.

Named Entity Recognition puede ser muy desafiante:

Back in 2000 , People Magazine PUBLISHER highlighted Prince Williams' PERSON style who at the time was a little more fashion-conscious , even making fashion statements at times .

Now-a-days the prince mainly wears navy COLOR suits ITEM (sometimes double-breasted DESIGN) , light blue COLOR button-ups ITEM with classic LOOK pointed DESIGN collars PART , and burgundy COLOR ties ITEM .

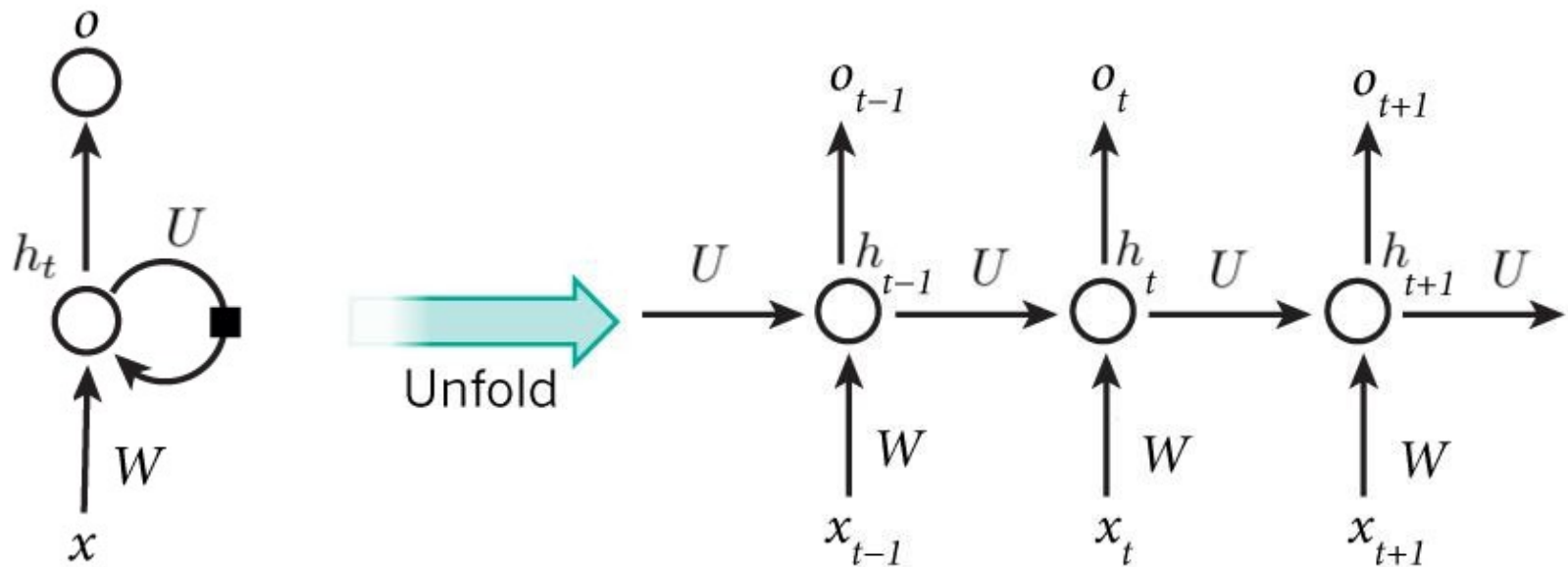
But who knows what the future holds ...

Duchess Kate PERSON did wear an Alexander McQueen BRAND dress ITEM to the wedding OCCASION in the fall of 2017 SEASON .

Sequence labeling

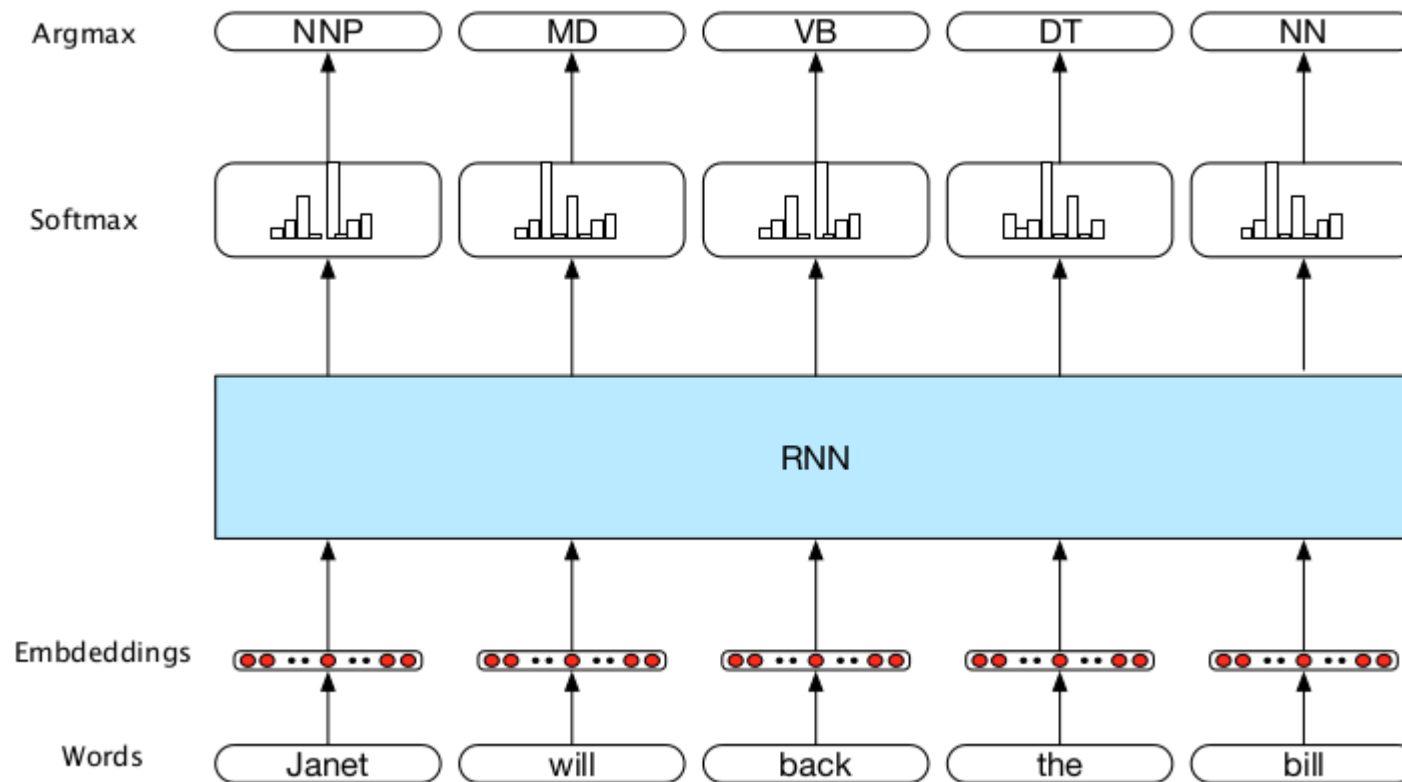
Presentamos los datos como secuencia: $X = (x_1, x_2, \dots, x_T)$

Recurrente convencional: $h_t = g(W \cdot x_t + U \cdot h_{t-1})$



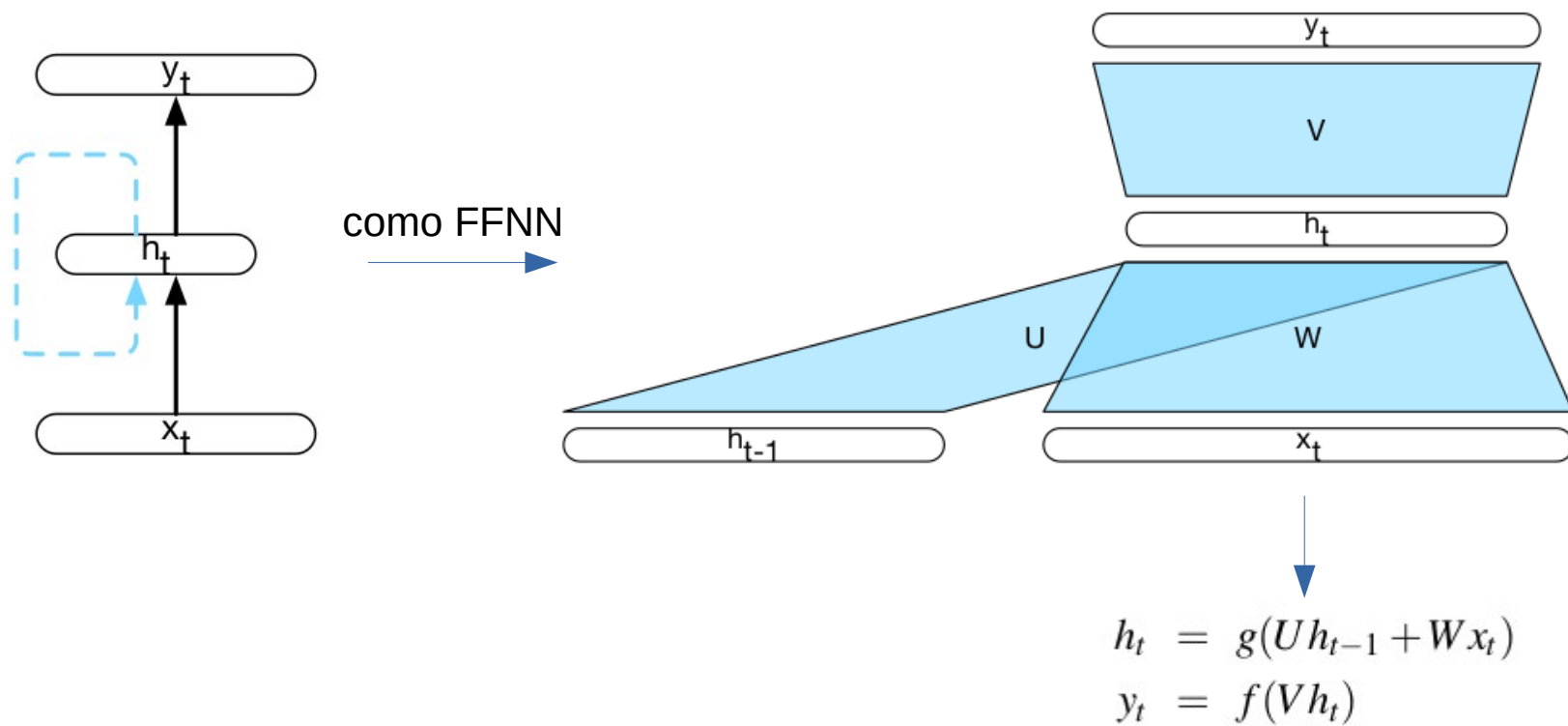
Sequence labeling

Red recurrente (**sequence labeling**):



Sequence labeling

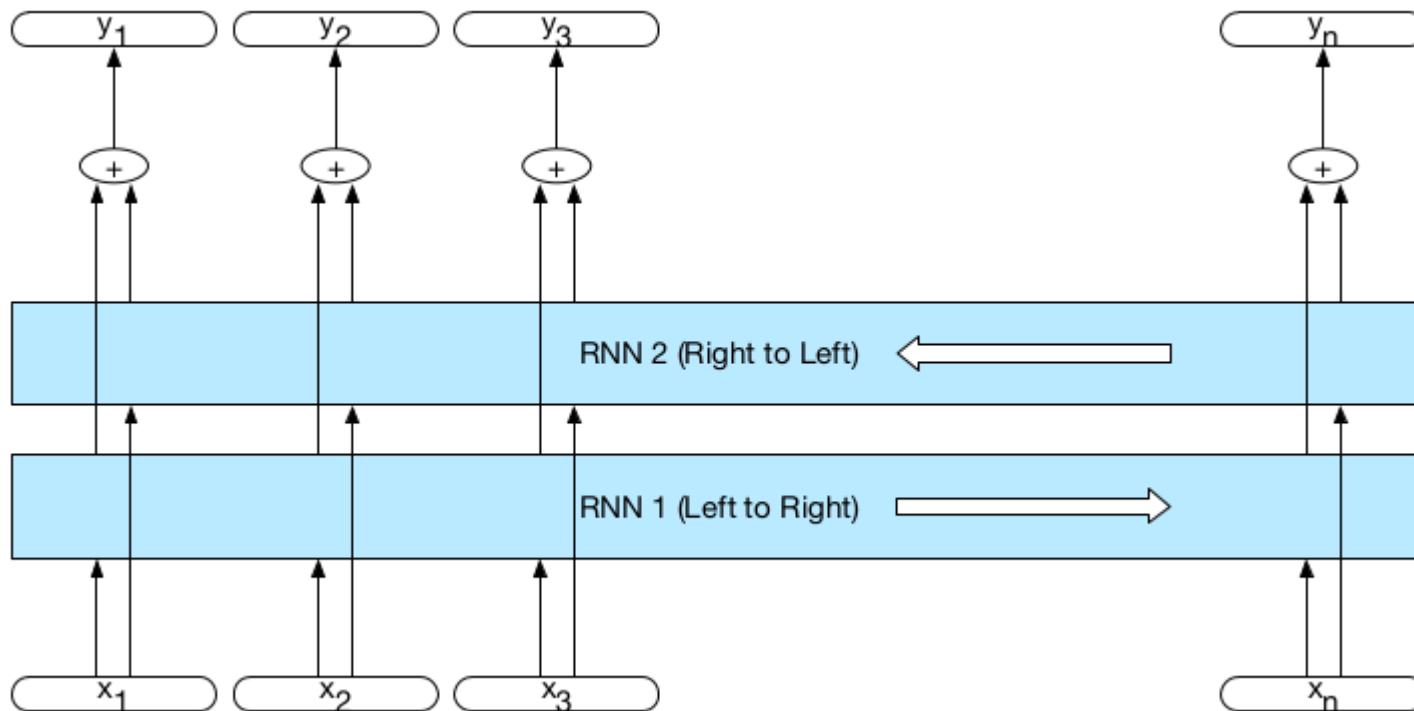
Red recurrente:



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

Sequence labeling

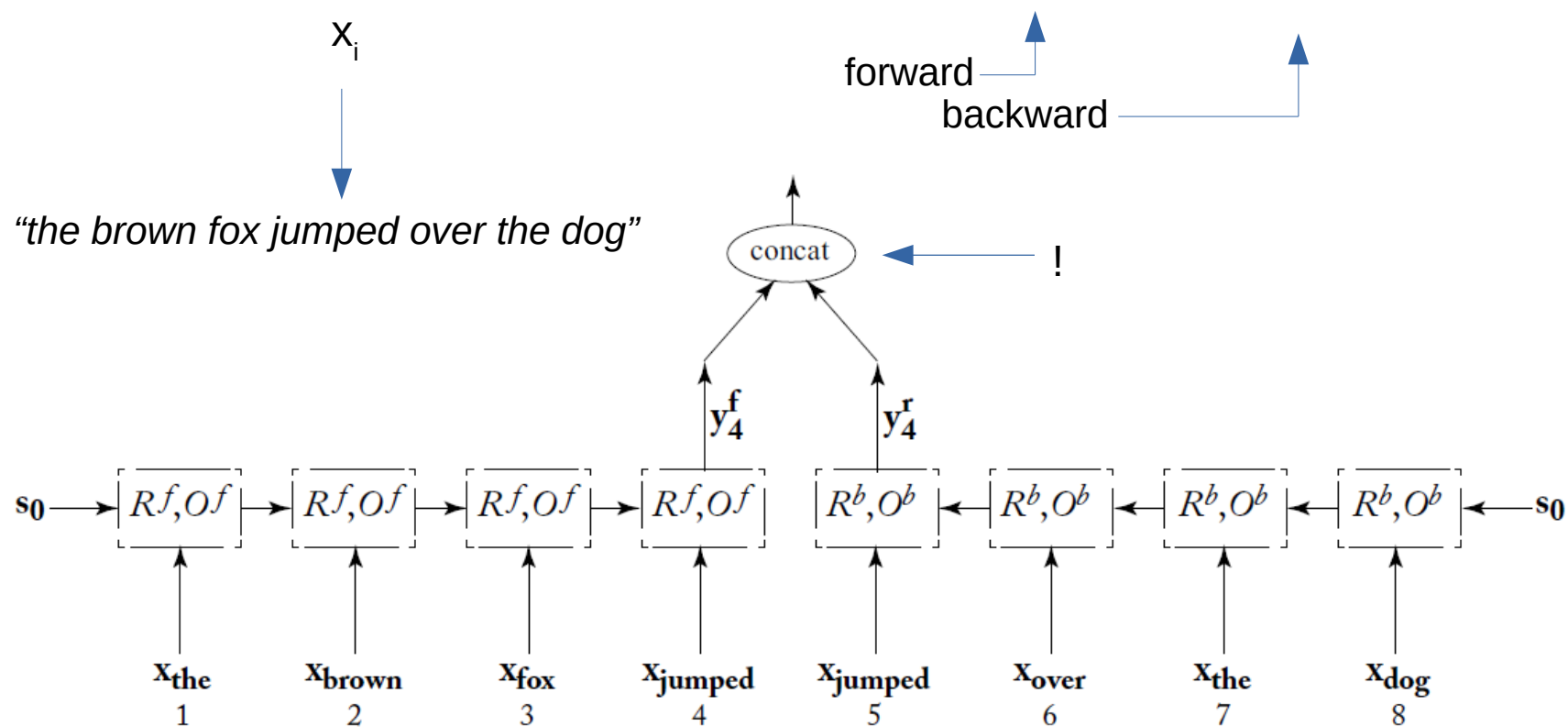
Bidirectional RNN (biRNN):



Sequence labeling

Bidirectional RNN (biRNN):

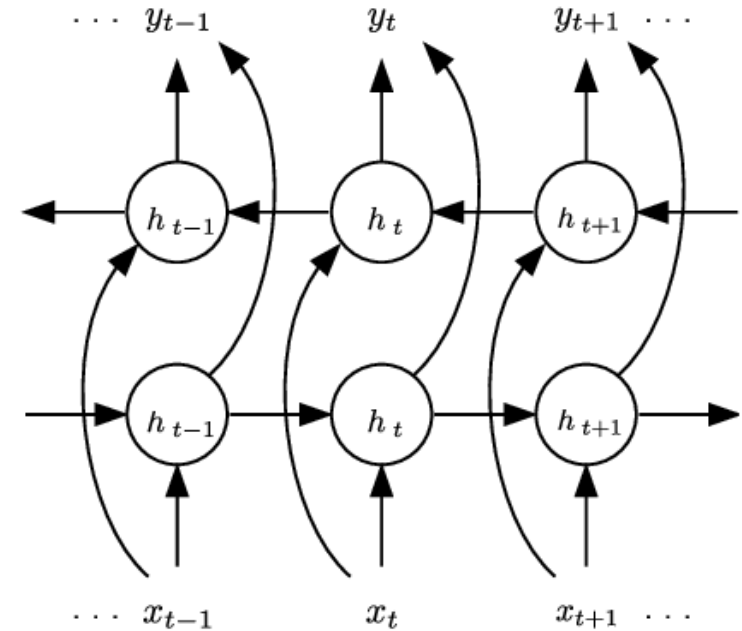
$$\text{biRNN}(x_{1:n}, i) = y_i = [\text{RNN}^f(x_{1:i}); \text{RNN}^b(x_{n:i})].$$



Sequence labeling

Bidirectional RNN (biRNN):

$$\begin{aligned}h_t^{(f)} &= g(W^{(f)} \cdot x_t + U^{(f)} \cdot h_{t-1}^{(f)}) \\h_t^{(b)} &= g(W^{(b)} \cdot x_t + U^{(b)} \cdot h_{t+1}^{(b)}) \\y_t &= g(V^{(f)} \cdot h_t^{(f)} + V^{(b)} \cdot h_t^{(b)})\end{aligned}$$



Sequence labeling

Bidirectional RNN para POS tagging con modelo preentrenado de subpalabras

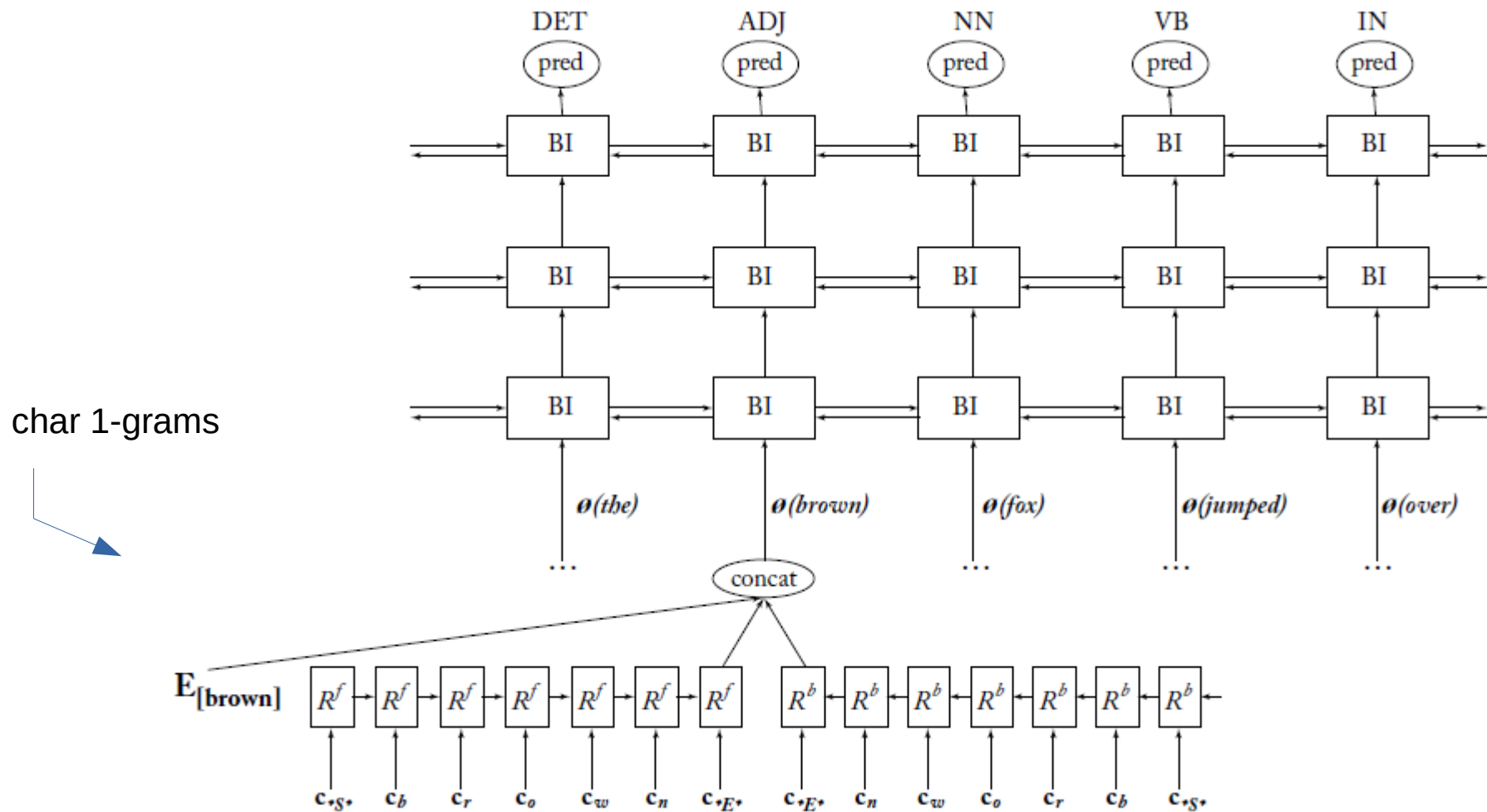
Char embeddings: $x_i = \phi(s, i) = [E_{[w_i]}; \text{RNN}^f(c_{1:\ell}); \text{RNN}^b(c_{\ell:1})]$.

FastText  char n-grams

POS-tagging: $p(t_i = j | w_1, \dots, w_n) = \text{softmax}(\text{MLP}(\text{biRNN}(x_{1:n}, i)))_{[j]}$

Sequence labeling

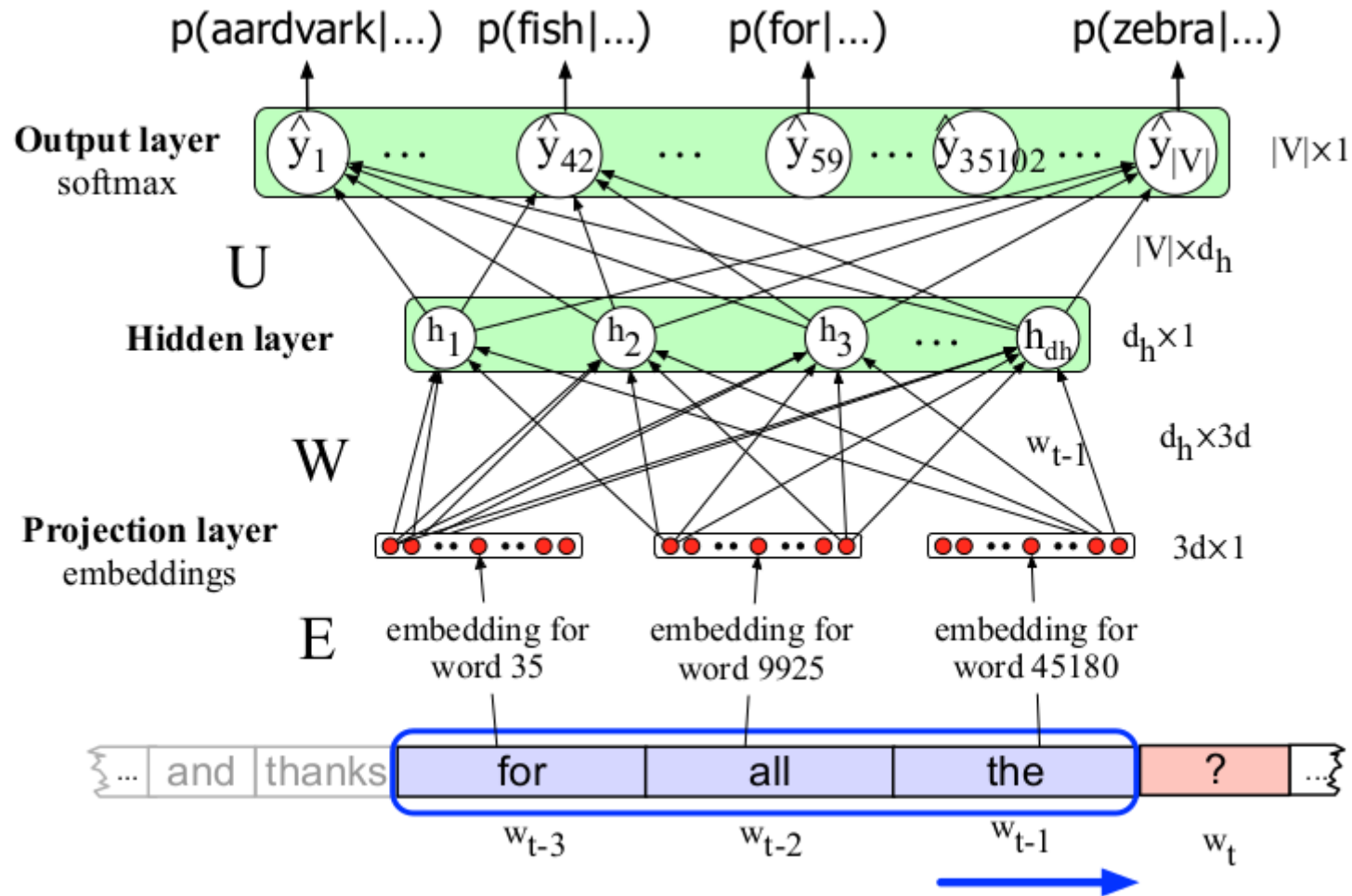
Bidirectional RNN para POS tagging con modelo preentrenado de subpalabras



- MODELOS DE LENGUAJE -

Background: Modelos de lenguaje neuronales

Feed-forward (modelo de lenguaje): **token prediction**



Background: Modelos de lenguaje neuronales

Feed-forward (modelo de lenguaje): **token prediction**

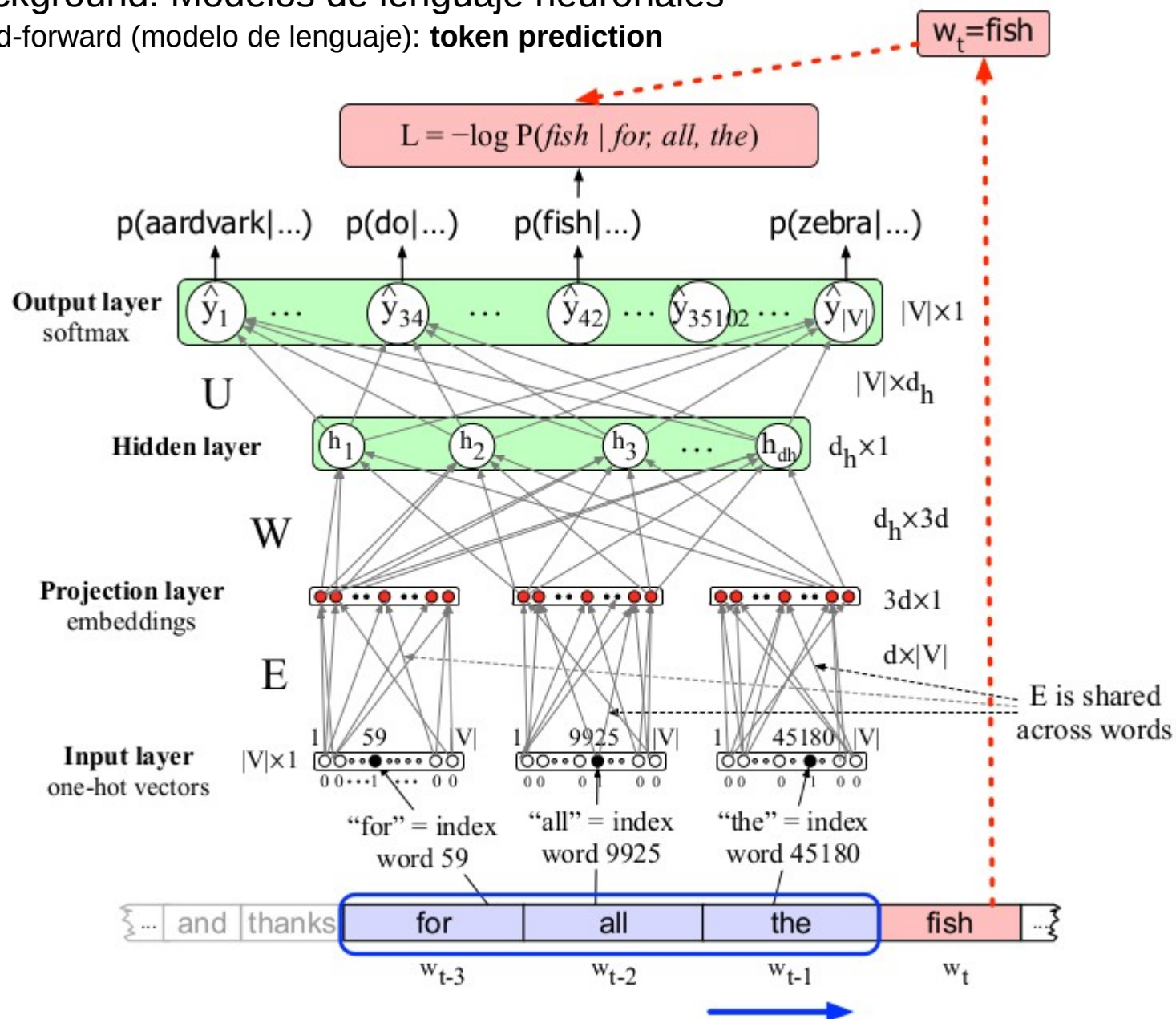
Task!!!

$$e = (Ex_1, Ex_2, \dots, Ex)$$

$$h = \sigma(We + b)$$

$$z = Uh$$

$$\hat{y} = \text{softmax}(z)$$



Background: Modelos de lenguaje neuronales

Red recurrente:

output distribution

$$\hat{y}^{(t)} = \text{softmax} \left(U h^{(t)} + b_2 \right) \in \mathbb{R}^{|V|}$$

hidden states

$$h^{(t)} = \sigma \left(W_h h^{(t-1)} + W_e e^{(t)} + b_1 \right)$$

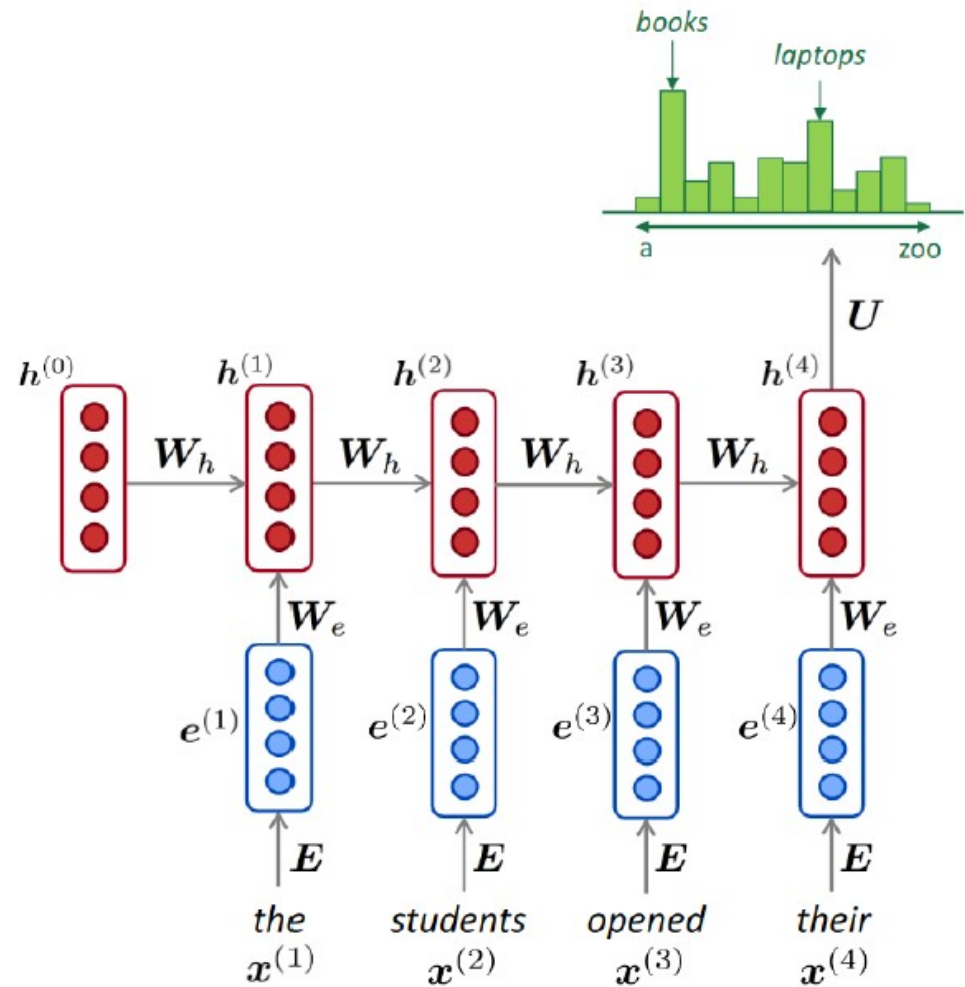
$h^{(0)}$ is the initial hidden state

word embeddings

$$e^{(t)} = E x^{(t)}$$

words / one-hot vectors

$$x^{(t)} \in \mathbb{R}^{|V|}$$



Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C. A Neural Probabilistic Language Model. *Journal of Machine Learning Research* 3:1137-1155, 2003.

Background: Modelos de lenguaje neuronales

Red recurrente:

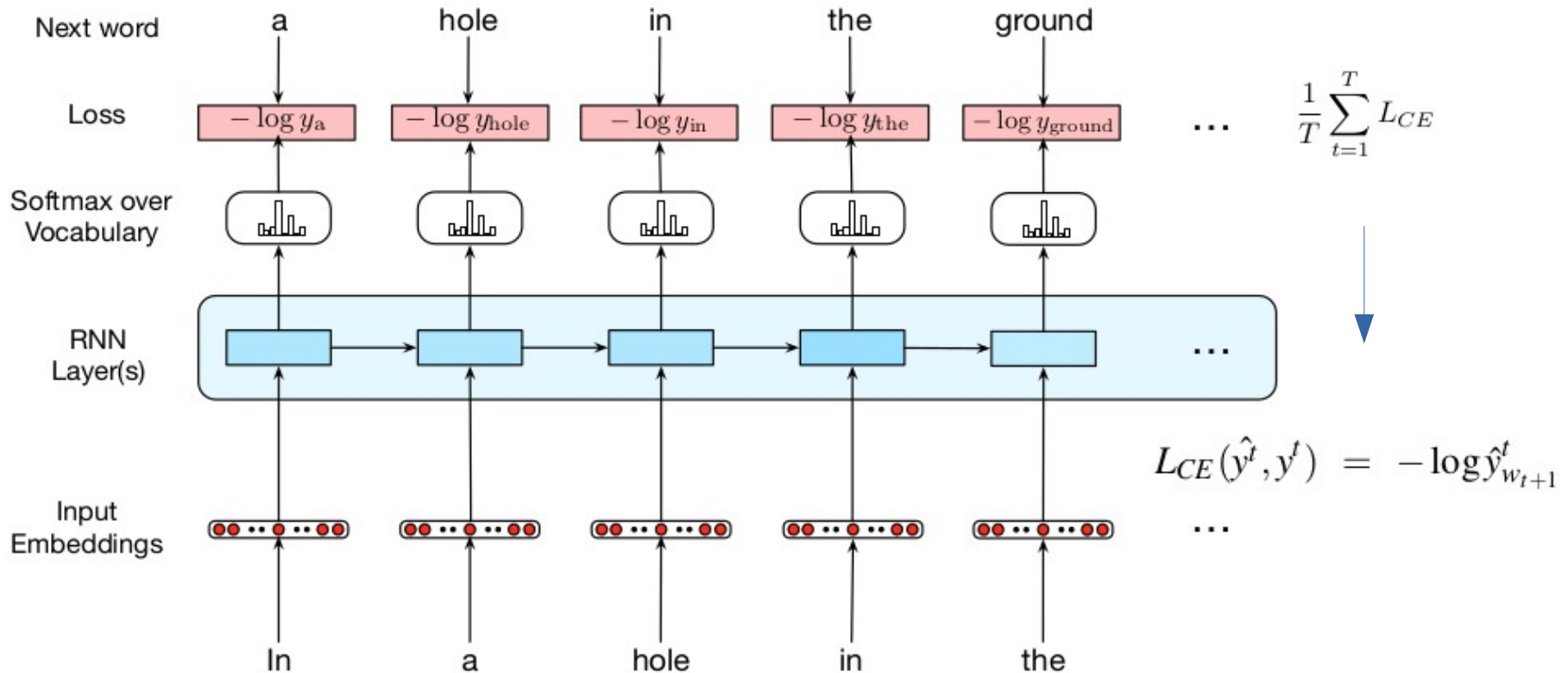
$$e_t = E^T x_t$$

$$h_t = g(Uh_{t-1} + We_t)$$

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

Entrenamiento auto-regresivo:

$$P(w_{t+1} = i | w_{1:t}) = y_t^i$$



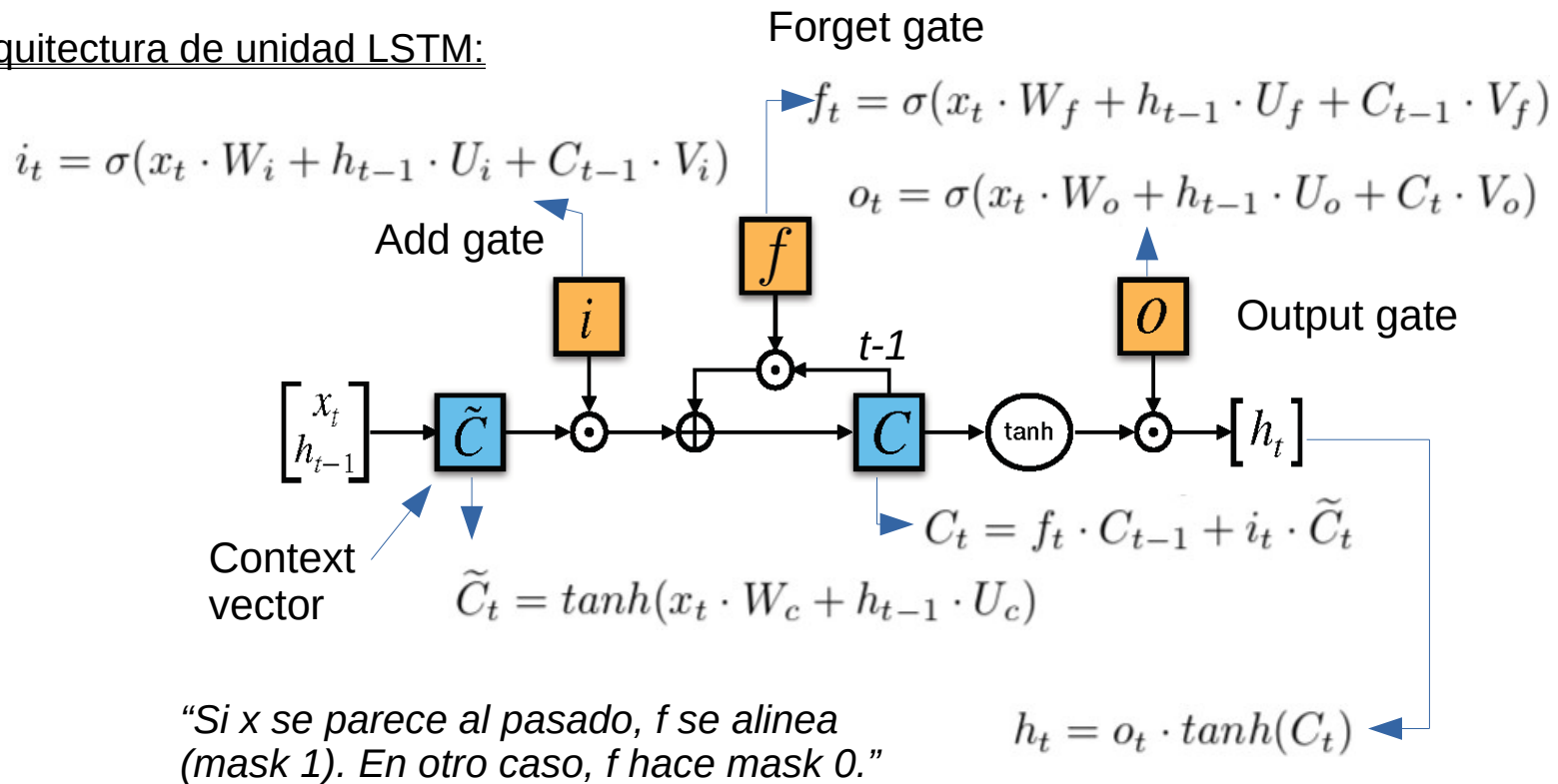
- DEPENDENCIA DEL CONTEXTO -

Extensión a LSTM

Long-short term memory (LSTM): dividen el problema en dos sub-problemas: a) remover información innecesaria, b) agregar información necesaria para resolver una tarea.

Las LSTM hacen lo anterior agregando unidades de control de flujo de información. Cada unidad consiste de una capa FF seguida por una sigmoid (activación). Combinando la salida de la sigmoid con productos, se obtiene un efecto similar al de una máscara binaria.

Arquitectura de unidad LSTM:



Dependencia del contexto

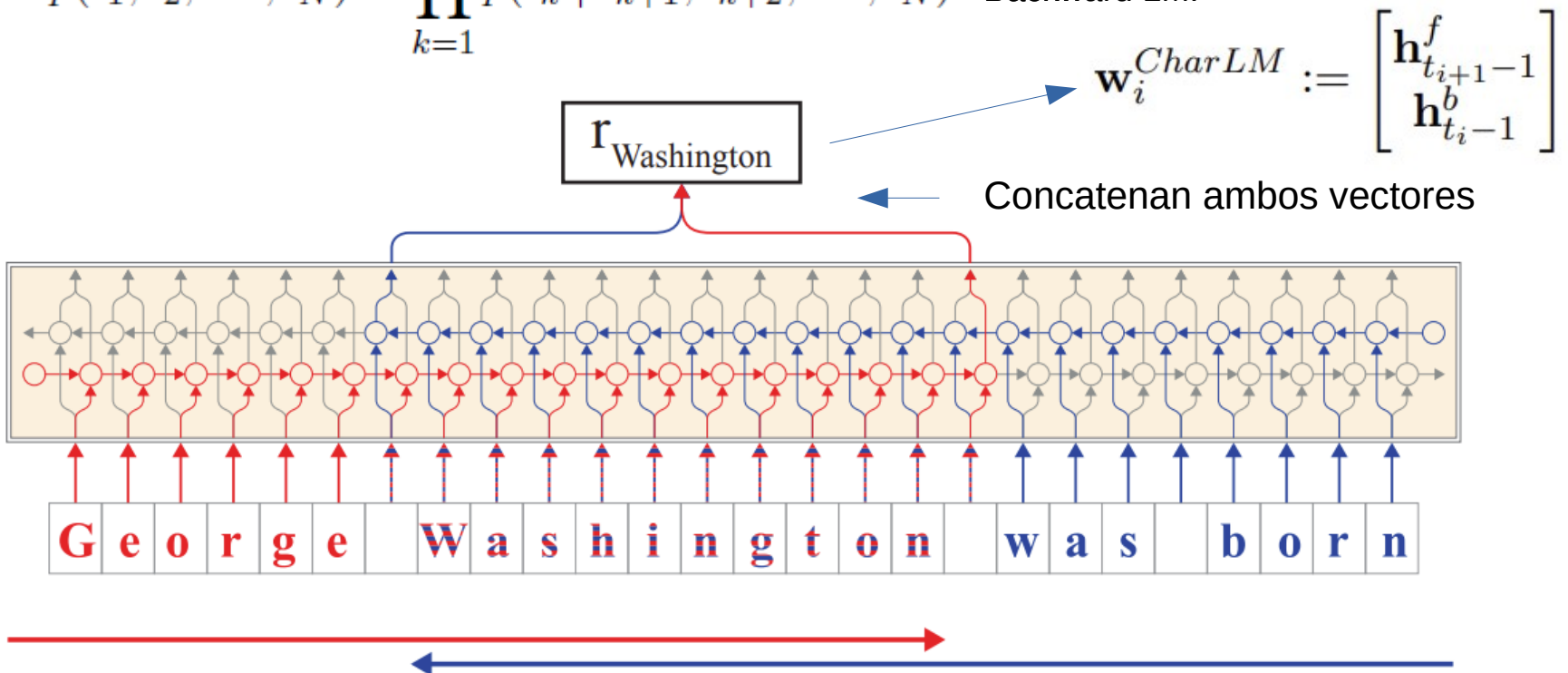
FLAIR

Dada una secuencia de n símbolos, se calculan dos modelos de lenguajes:

LSTM

$$p(t_1, t_2, \dots, t_N) = \prod_{k=1}^N p(t_k \mid t_1, t_2, \dots, t_{k-1}). \quad \text{Forward LM.}$$

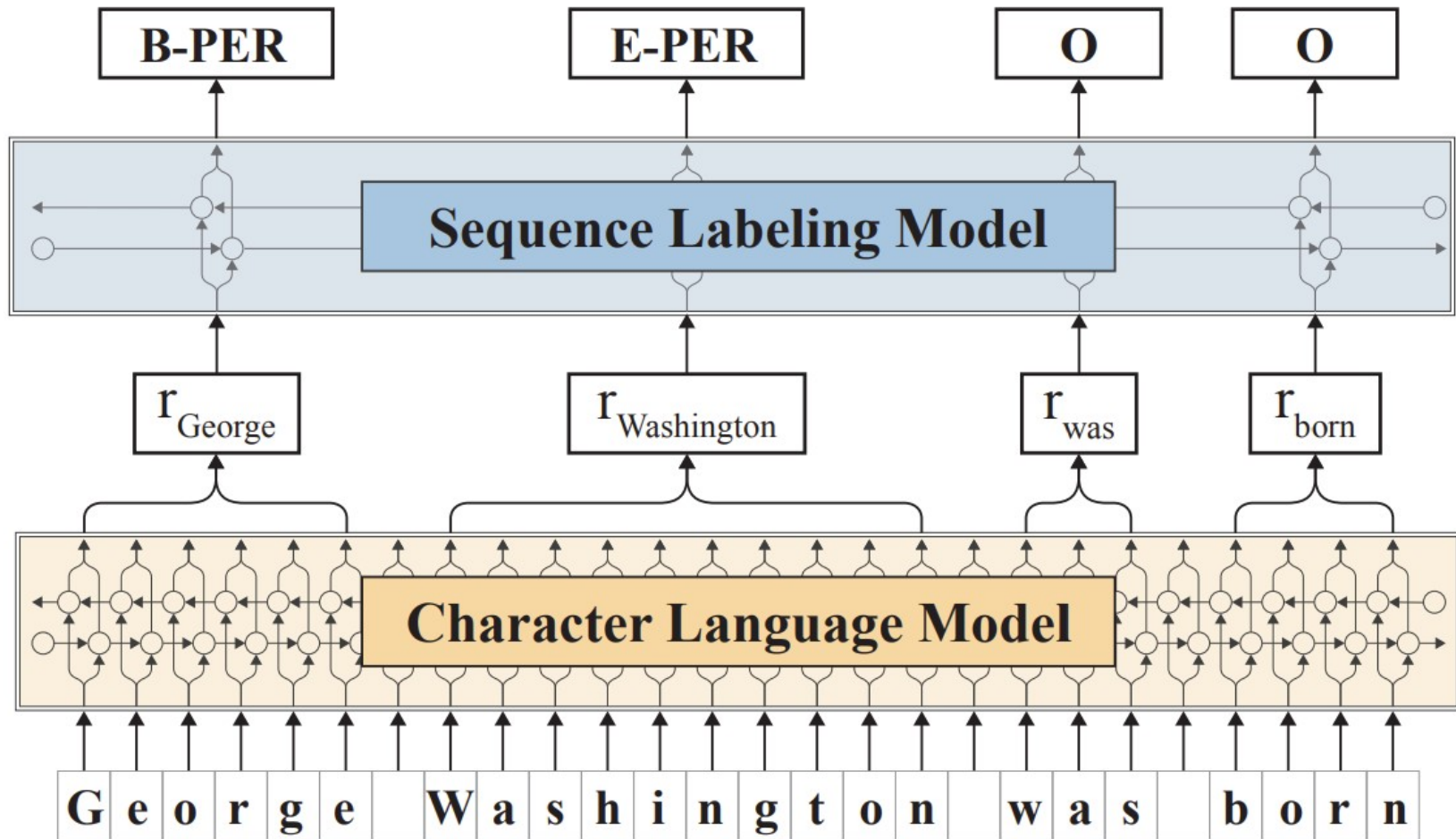
$$p(t_1, t_2, \dots, t_N) = \prod_{k=1}^N p(t_k \mid t_{k+1}, t_{k+2}, \dots, t_N). \quad \text{Backward LM.}$$



Dependencia del contexto

FLAIR

Luego usan los *string embeddings* en NER:



Dependencia del contexto

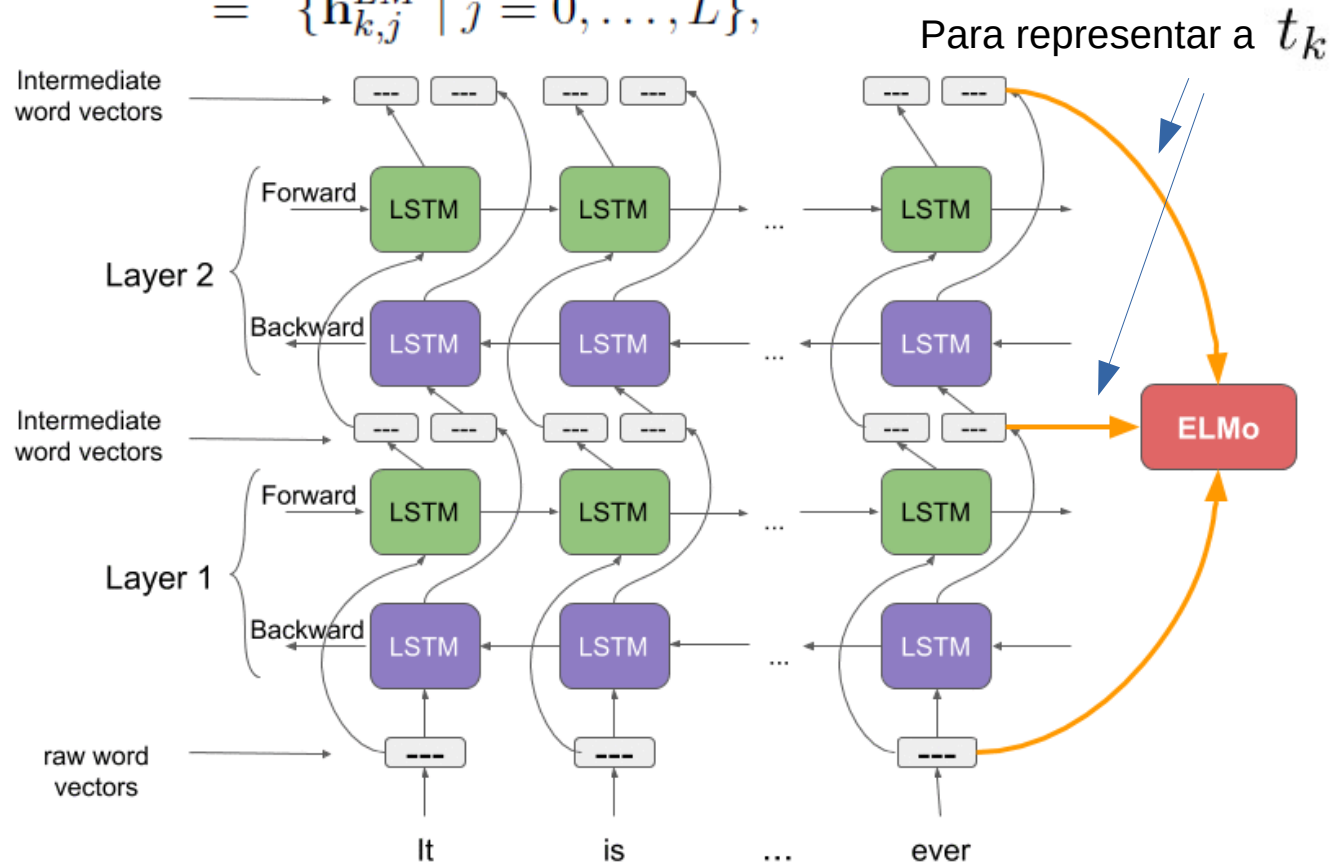
Embeddings basados en modelos de lenguaje (ELMo)

Idea similar a FLAIR pero con multicapa y modelos basados en palabras:

Parámetros de la representación \leftarrow \rightarrow Parámetros de las LSTM

$$\begin{aligned} R_k &= \{\mathbf{x}_k^{LM}, \vec{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \dots, L\} \\ &= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \dots, L\}, \end{aligned}$$

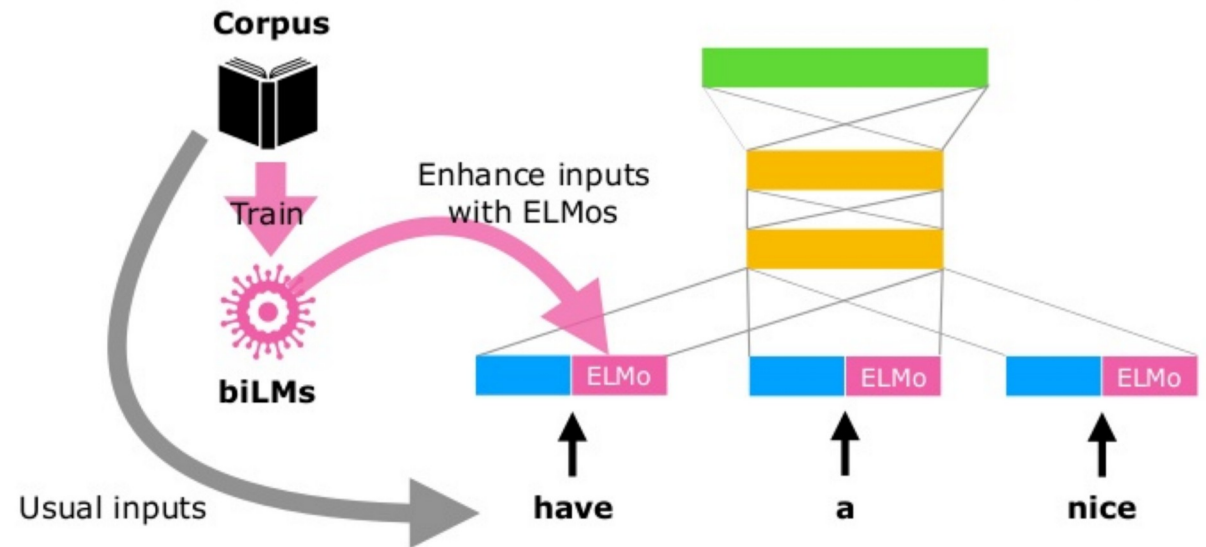
Arquitectura:



Dependencia del contexto

Embeddings basados en modelos de lenguaje (ELMo)

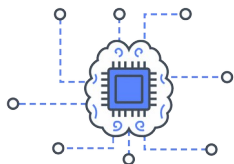
ELMo: se puede combinar con otras representaciones según la tarea específica:



El modelo final de ELMo (pre-entrenado) usa dos capas LSTM con vectores de dimensionalidad 512. Para la capa de representación, ELMo usa char n-grams y una capa de proyección de dimensionalidad 512.

ELMo fue entrenado sobre el **1 Billion Word Benchmark**.

Fine tuning para downstream tasks implica disminución en *perplexity* y mejoras en desempeño.



Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In INTERSPEECH.