1.2 MODELOS GRAMATICALES Y OTROS

MODELOS GRAMATICALES Y OTROS

- Introducción
- Modelos incontextuales
- Otros modelos

MODELOS GRAMATICALES Y OTROS

- Introducción
- Modelos incontextuales
- Otros modelos

Introducción

Gramáticas tipo 2 (incontextuales)

Tienen su origen principalmente en trabajos de Procesamiento del Lenguaje Natural. Además de las gramáticas incontextuales, se han utilizado gramáticas LR, gramáticas de unificación, etc.

Principales ventajas:

- La representación del lenguaje es potente, tan cercana al lenguaje natural como se “quiera”.

Principales inconvenientes:

- Presentan métodos de análisis computacionalmente costosos .
- Su integración con los modelos acústicos es computacionalmente costosa.
- Está poco explorada la posibilidad de su estimación automática a partir de muestras.

Introducción

Gramáticas tipo 3 (regulares)

Gramáticas regulares y autómatas finitos.

Principales ventajas:

- Presentan una sencilla implementación, con algoritmos de análisis de coste lineal.
- Permiten una sencilla integración con los modelos acústicos.
- Permiten la aplicación de técnicas de Inferencia Gramatical para la estimación de los modelos.

Principales inconvenientes:

- Su potencia de representación del lenguaje puede resultar débil para tareas complejas.

MODELOS GRAMATICALES

- Introducción
- Modelos incontextuales
- Otros modelos

Modelos Incontextuales

Gramáticas de propósito general

- Algoritmo de Earley
- Algoritmo de Cocke-Younger-Kasami

Gramáticas LR

- Extensión del análisis LR clásico para el tratamiento de gramáticas quasi-LR, con mínima pérdida de eficiencia.

  Incorpora un método para tratar las entradas múltiples en la tabla de análisis.

  Adapta su algoritmo para una entrada en forma de lattice.

Modelos Incontextuales

El modelo formal más comúnmente usado para representar la estructura de constituyentes de los lenguajes naturales es la **Gramática Incontextual** (llamada también Phrase-structure grammar y equivalente a la conocida como Backus-Naur form).

Sea G=(N, Σ, R, S) una gramática formal

- N es el conjunto de símbolos no terminales (componentes y categorías sintácticas)
- Σ es el conjunto de símbolos terminales (palabras)
- R es el conjunto de reglas de producción: α→β con $\alpha \in (N \cup \Sigma)^* N (N \cup \Sigma)^*$
- S ∈ N

Es **incontextual** cuando todas sus reglas presentan la forma

$$A \to \alpha \quad A \in N \quad \alpha \in (N \cup \Sigma)^*$$

Está en **Forma Normal de Chomsky** (FNC) cuando todas sus reglas presentan la forma

$$A \to BC \quad A, B, C \in N \quad o \quad A \to a \quad a \in \Sigma$$

Modelos Incontextuales

Una gramática para el lenguaje del corpus ATIS

Noun	→	*flights* | *breeze* | *trip* | *morning* | ...
Verb	→	*is* | *prefer* | *like* | *need* | *want* | *fly*
Adjective	→	*cheapest* | *non – stop* | *first* | *latest* | *other* | *direct* | ...
Pronoun	→	*me* | *I* | *you* | *it* | ...
Proper-Noun	→	*Alaska* | *Baltimore* | *Los Angeles* | *Chicago* | *United* | *American* | ...
Determiner	→	*the* | *a* | *an* | *this* | *these* | *that* | ...
Preposition	→	*from* | *to* | *on* | *near* | ...
Conjunction	→	*and* | *or* | *but* | ...

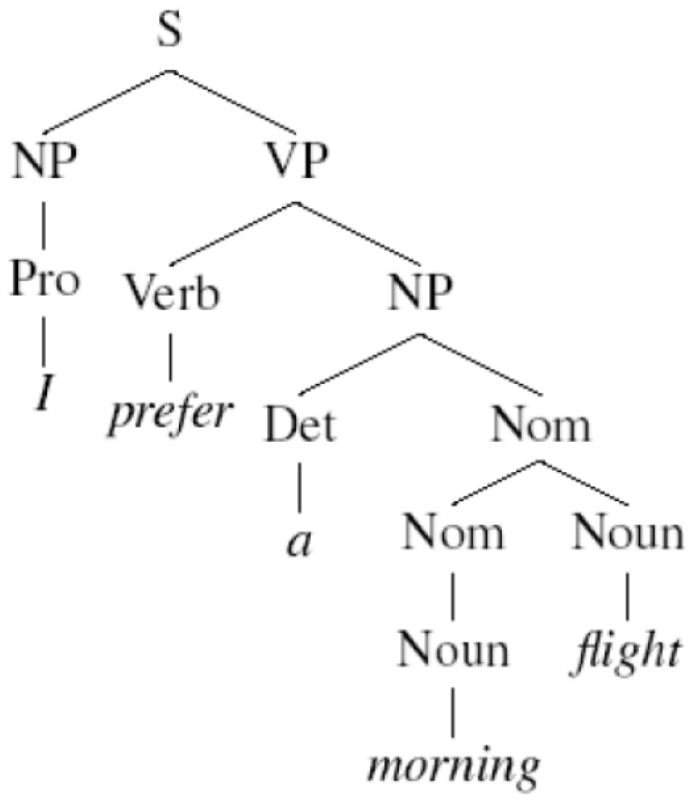
Modelos Incontextuales

Una gramática para el lenguaje del corpus ATIS

S	$\rightarrow$	*NP VP*	I + want a morning flight
NP	$\rightarrow$	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	$\rightarrow$	*Nominal Noun*	morning + flight
	|	*Noun*	flights
VP	$\rightarrow$	*Verb*	do
	|	*Verb NP*	want + a flight
	|	*Verb NP PP*	leave + Boston + in the morning
	|	*Verb PP*	leaving + on Thursday
PP	$\rightarrow$	*Preposition NP*	from + Los Angeles

Modelos Incontextuales

Un árbol de derivación para la frase: “I prefer a morning flight”



En notación parentizada este árbol de derivación es:

[S [NP [Pro I]] [VP [Verb prefer] [NP [Det a] [Nom [Nom [Noun morning]] [Noun flight]]]]]

Modelos Incontextuales

Dada una gramática incontextual probabilística $G=(N,\Sigma,R,S,P)$ donde P es el conjunto de probabilidades asociadas a las reglas de la gramática R, se dice que es ***estocástica*** si cumple la siguiente condición de normalización para todo $A \in N$:

$$\sum_{\forall \alpha \in (N \cup \Sigma)^*} P(A \rightarrow \alpha) = 1$$

Decimos que una gramática incontextual estocástica (GIE) G es consistente cuando la probabilidad total de las cadenas de $L(G)$ es igual a 1, es decir

$$\sum_{\forall w \in \Sigma^*} P(S \overset{*}{\Rightarrow} w) = 1$$

(Vamos a suponer que trabajamos con gramáticas consistentes).

Modelos Incontextuales

Los problemas clásicos

Sea la GIE $G=(N,\Sigma,R,S,P)$, la solución a los siguientes problemas son de interés en la Modelización del Lenguaje:

1. ¿Cuál es la probabilidad con la que la gramática genera una determinada cadena $w \in \Sigma^*$?
2. ¿Cuál es la derivación más probable de una determinada cadena $w \in \Sigma^*$ (aquella en la cual el producto de las probabilidades de sus reglas es máximo), y cuál es la probabilidad de dicha derivación?
3. ¿Dada una cadena $w \in \Sigma^*$, cuál es la probabilidad de que la gramática genere una cadena que contiene como prefijo a w?
4. Dado el conjunto de reglas de la gramática R, ¿cuál es el conjunto de probabilidades asociadas P?

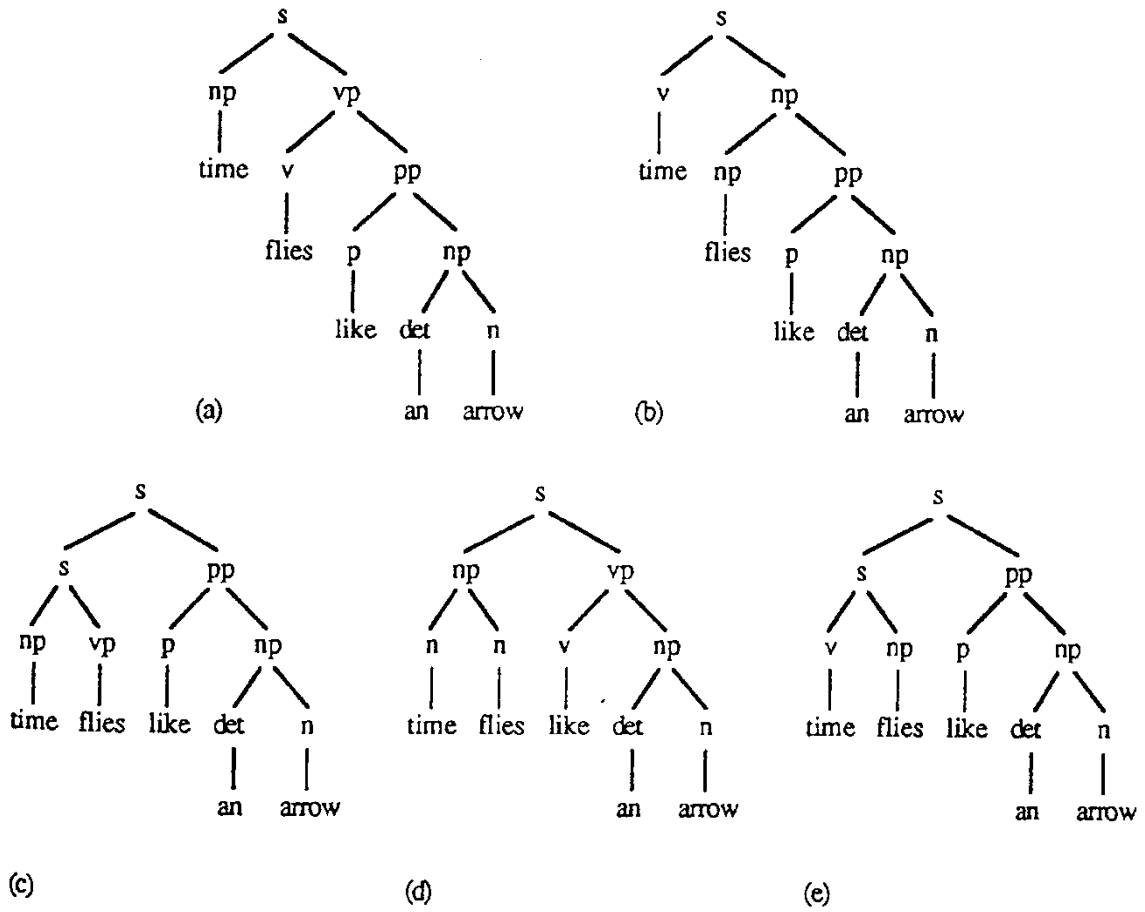
Modelos Incontextuales

(s - 1)	s	→	np	vp	P(1|s)	= .5
(s - 2)	s	→	s	pp	P(2|s)	= .25
(s - 3)	s	→	v	np	P(3|s)	= .05
(s - 4)	s	→	v	pp	P(4|s)	= .1
(s - 5)	s	→	flies		P(5|s)	= .04
(s - 6)	s	→	like		P(6|s)	= .04
(s - 7)	s	→	time		P(7|s)	= .02
(vp - 1)	vp	→	v	np	P(1|vp)	= .2
(vp - 2)	vp	→	v	pp	P(2|vp)	= .4
(vp - 3)	vp	→	flies		P(3|vp)	= .16
(vp - 4)	vp	→	like		P(4|vp)	= .16
(vp - 5)	vp	→	time		P(5|vp)	= .08
(np - 1)	np	→	det	n	P(1|np)	= .4
(np - 2)	np	→	n	n	P(2|np)	= .1
(np - 3)	np	→	np	pp	P(3|np)	= .1
(np - 4)	np	→	arrow		P(4|np)	= .16
(np - 5)	np	→	flies		P(5|np)	= .08
(np - 6)	np	→	time		P(6|np)	= .16
(pp - 1)	pp	→	p	np	P(1|pp)	= 1.0
(n - 1)	n	→	arrow		P(1|n)	= .4
(n - 2)	n	→	flies		P(2|n)	= .2
(n - 3)	n	→	time		P(3|n)	= .4
(v - 1)	v	→	flies		P(1|v)	= .4
(v - 2)	v	→	like		P(2|v)	= .4
(v - 3)	v	→	time		P(3|v)	= .2
(p - 1)	p	→	like		P(1|p)	= 1.0
(det - 1)	det	→	an		P(1|det)	= 1.0

A probabilistic context-free grammar in Chomsky normal form

Modelos Incontextuales

Cinco derivaciones de la frase "time flies like an arrow"



Modelos Incontextuales

Algoritmo COCKE-YOUNGER-KASAMI

ENTRADA: G=(N,Σ,R,S,) una Gramática Incontextual en FNC,
$w \in \Sigma^*$ una cadena ($w=w_1 w_2 ... w_n$)

SALIDA: Tabla de análisis (diagonal de n x n) en la cual cada posición <i,j> contiene

$$\{A \in N \mid A \overset{+}{\Rightarrow} w_{ij}\}$$

donde w_{ij} es la subcadena de w que va de la posición i a la j.

MÉTODO: por inducción sobre la longitud de w_{ij} :

$|w_{ij}|=1, \quad A \overset{+}{\Rightarrow} w_{ii}=w_i$ sii $A \to w_i \in R$

$|w_{ij}|>1, \quad A \overset{+}{\Rightarrow} w_{ij} = w_i w_{i+1} ... w_j$ sii existe $A \to BC \in R \wedge$ existe k, i<=k<j :
$B \overset{+}{\Rightarrow} w_{ik}, \quad C \overset{+}{\Rightarrow} w_{k+1j}$

Si la posición <1,n> contiene el axioma de la gramática, S, entonces la gramática genera la cadena w.

Modelos Incontextuales

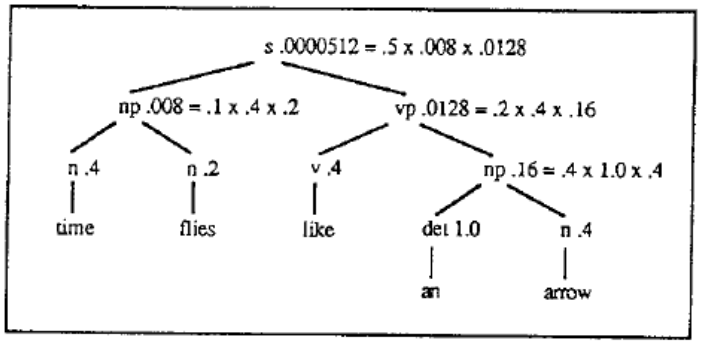
Algoritmo INSIDE

Entrada: G=(N,Σ,R,S,P) una GIE en FNC, y w∈Σ* una cadena.

Salida: P($S \overset{*}{\Rightarrow} w$)

Sea G=(N,Σ,R,S,P) una GIE, y sea w∈Σ* una cadena, $w=w_1w_2...w_K$. La probabilidad de la derivación $S \overset{*}{\Rightarrow} w$ es igual a la suma de las probabilidades de todos los árboles de derivación que generan w.

Su cálculo se basará en las probabilidades de $H \overset{*}{\Rightarrow} w_i...w_{i+n}\ \forall H \in N, i \geq 1, n \geq 0$. Se puede demostrar que en una GIE la probabilidad de las regla $H \rightarrow G_1G_2$ es igual al producto de las probabilidades de los dos subárboles con raíz G_1 y G_2 respectivamente, por la probabilidad de dicha regla.



Probabilities of a derivation tree

La base del algoritmo Inside viene dada por la siguiente ecuación:

$$P(H \overset{*}{\Rightarrow} w_i...w_{i+n}) = \sum_{G_1,G_2 \in N} P(H \rightarrow G_1G_2)[P(G_1 \overset{*}{\Rightarrow} w_i)P(G_2 \overset{*}{\Rightarrow} w_{i+1}...w_{i+n}) +$$

$$P(G_1 \overset{*}{\Rightarrow} w_iw_{i+1})P(G_2 \overset{*}{\Rightarrow} w_{i+2}...w_{i+n}) + ... +$$

$$P(G_1 \overset{*}{\Rightarrow} w_i...w_{i+n-1})P(G_2 \overset{*}{\Rightarrow} w_{i+n})]$$

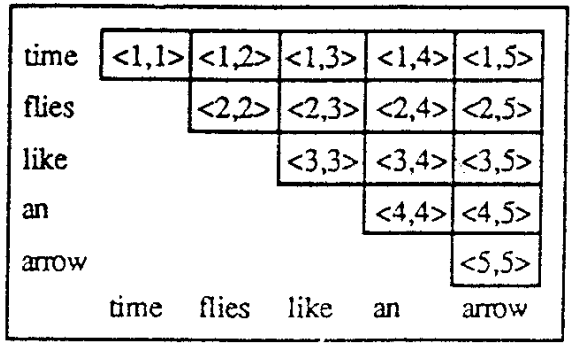
Modelos Incontextuales

Algoritmo INSIDE

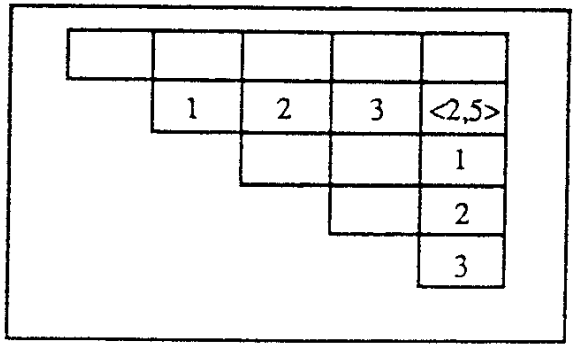
Un procedimiento eficiente para el cálculo de estas probabilidades utiliza una estructura de tabla propuesta en el algoritmo de Cocke-Younger-Kasami para el análisis de cadenas en gramáticas no probabilísticas. La posición <i,j> representa la subcadena $w_i...w_j$.

El cálculo de la probabilidad de que un no terminal H expanda la subcadena correspondiente a la posición <i,j> requiere el correspondiente cálculo de las posiciones a la izquierda y debajo de la posición <i,j>.

Por tanto, la tabla se rellena una diagonal tras otra, empezando por la más larga y acabando por el vértice (estrategia bottom-up).



The parse triangle

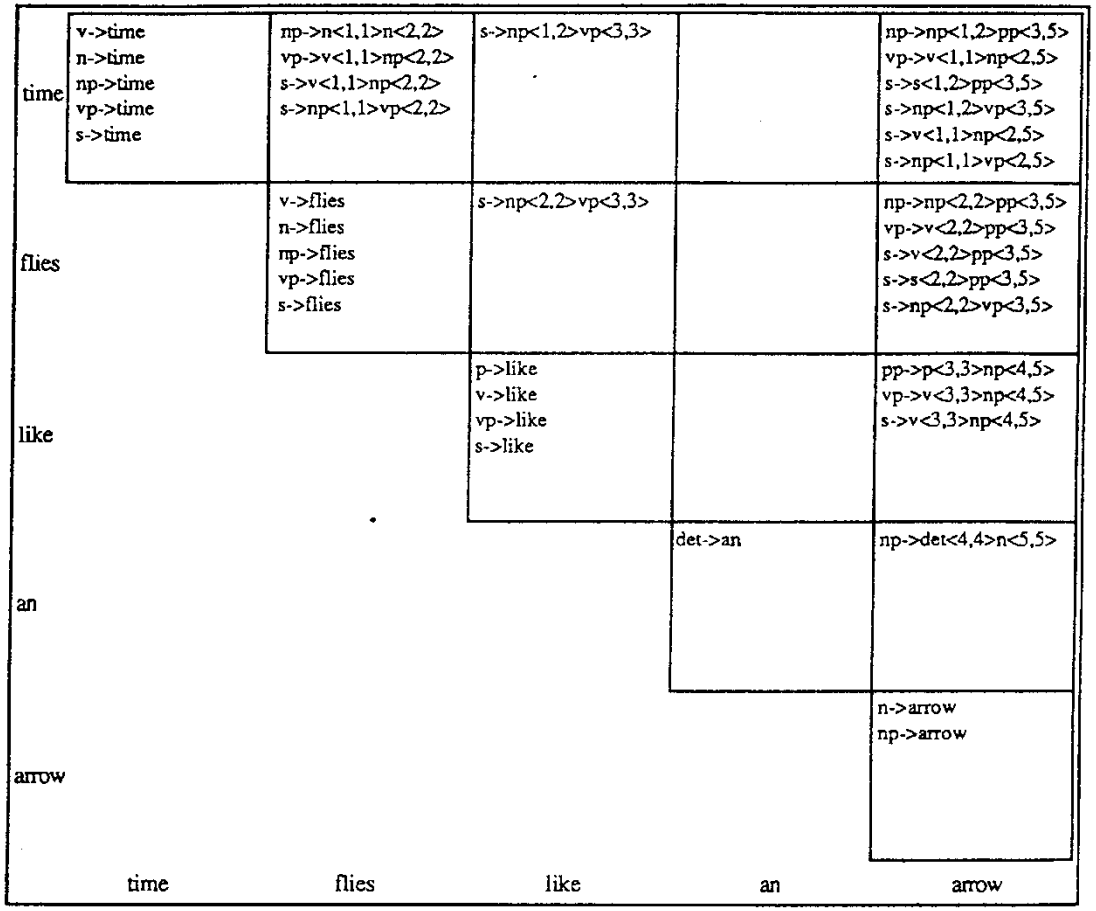


Contributions to cell <2,5>

Modelos Incontextuales

Algoritmo INSIDE

Tabla resultado de la aplicación del algoritmo INSIDE a la frase: “time flies like un arrow”.



The complete parse triangle for the sentence *"Time flies like an arrow"*

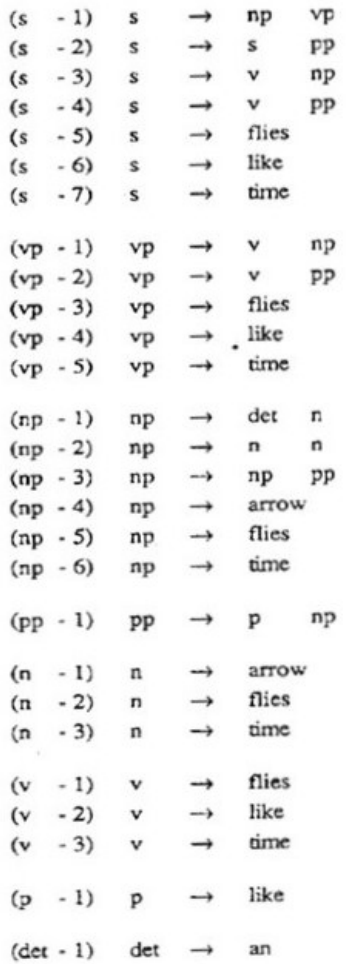
Algoritmo INSIDE

<1,1>

s -> time	(s-7)	P= 0,02
vp -> time	(vp-5)	P=0,08
np -> time	(np-6)	P=0,16
n -> time	(n-3)	P=0,4
v -> time	(v-3)	P=0,2

<2,2>

s -> flies	(s-5)	P=0,04
vp -> flies	(vp-3)	P=0,16
np -> flies	(np-5)	P=0,08
n -> flies	(n-2)	P=0,2
v -> flies	(v-1)	P=0,4



(s - 1) s → np vp
(s - 2) s → s pp
(s - 3) s → v np
(s - 4) s → v pp
(s - 5) s → flies
(s - 6) s → like
(s - 7) s → time

(vp - 1) vp → v np
(vp - 2) vp → v pp
(vp - 3) vp → flies
(vp - 4) vp → like
(vp - 5) vp → time

(np - 1) np → det n
(np - 2) np → n n
(np - 3) np → np pp
(np - 4) np → arrow
(np - 5) np → flies
(np - 6) np → time

(pp - 1) pp → p np

(n - 1) n → arrow
(n - 2) n → flies
(n - 3) n → time

(v - 1) v → flies
(v - 2) v → like
(v - 3) v → time

(p - 1) p → like

(det - 1) det → an

Algoritmo INSIDE

<1,1>

s -> time
vp -> time
np -> time
n -> time
v -> time

<2,2>

s -> flies
vp -> flies
np -> flies
n -> flies
v -> flies

<1,2>

s -> np vp	(s-1)
np -> n n	(np-2)
s -> v np	(s-3)
vp -> v np	(vp-1)

(s - 1)	s	→	np	vp
(s - 2)	s	→	s	pp
(s - 3)	s	→	v	np
(s - 4)	s	→	v	pp
(s - 5)	s	→	flies	
(s - 6)	s	→	like	
(s - 7)	s	→	time	
(vp - 1)	vp	→	v	np
(vp - 2)	vp	→	v	pp
(vp - 3)	vp	→	flies	
(vp - 4)	vp	→	like	
(vp - 5)	vp	→	time	
(np - 1)	np	→	det	n
(np - 2)	np	→	n	n
(np - 3)	np	→	np	pp
(np - 4)	np	→	arrow	
(np - 5)	np	→	flies	
(np - 6)	np	→	time	
(pp - 1)	pp	→	p	np
(n - 1)	n	→	arrow	
(n - 2)	n	→	flies	
(n - 3)	n	→	time	
(v - 1)	v	→	flies	
(v - 2)	v	→	like	
(v - 3)	v	→	time	
(p - 1)	p	→	like	
(det - 1)	det	→	an	

Modelos Incontextuales

Algoritmo VITERBI

ENTRADA: G=(N,Σ,R,S,P) una GIE, y una cadena w∈Σ*.

SALIDA: la probabilidad de la derivación de máxima probabilidad de w en G.

Sea T un árbol de derivación de una cadena, y sea U un subárbol de T con raíz H que expande la subcadena <i,j>. Si existe un subárbol U'≠U con la misma raíz H que expande la misma subcadena <i,j>, otro árbol de derivación T' se obtiene reemplazando U por U' en T.

El algoritmo de Viterbi se basa en la observación de que si P(U)>P(U') entonces P(T)>P(T').

El procedimiento para el algoritmo Inside se puede adaptar para el algoritmo de Viterbi: en cada posición, para cada no terminal H, de entre todas las probabilidades de todos los posibles subárboles con raíz H, se retiene la mayor, en lugar de la suma.

MODELOS GRAMATICALES

- Introducción
- Modelos regulares
- Modelos incontextuales
- Otros modelos

Otros: Skip-grams y Factored LM

- Skip-grams
  - Skip-grams es una técnica basada en n-gramas que, además de secuencias de palabras adyacentes, permite que algunas palabras sean “skipped”.
  - Dada una cierta distancia *k, permite un* total de *k* o menos skips para construir el n-gram. Por ejemplo: un “4-skip-n-gram” incluye 4 skips, 3 skips, 2 skips, 1 skip, y 0.

  Ejemplo de 2-skip-bi-grams comparado con el bigrama estándar.

  Sea la oración: “Insurgents killed in ongoing fighting.”

  Bi-grams = {insurgents killed, killed in, in ongoing, ongoing fighting}.

  2-skip-bi-grams = {insurgents killed, insurgents in, insurgents ongoing, killed in, killed ongoing, killed fighting, in ongoing, in fighting, ongoing fighting}

- Factored LM
  - En un *factored language model*, una palabra es vista como un vector de k factors.
  - Los factors pueden incluir conocimiento de todo tipo: clases morfológicas, stems, lemas, características de las lenguas fuertemente declinados (árabe, alemán, finés, etc.), clases, etc.

Modelos basados en Redes Neuronales

- **Modelos de lenguaje causales.**

- **Representación vectorial de palabras y textos.**

Modelos de lenguaje causales

- Como los modelos de N-gramas, asignan probabilidad a una secuencia de palabras, pero la distribución de probabilidad sobre el próximo token se calcula mediante una red neuronal.

- Procesan el texto unidireccionalmente y, dado un prefijo, calculan una distribución de probabilidad sobre el siguiente token.

- Si la red subyacente permite manejar secuencias, no es necesario considerar la asunción de Markov (no se olvida el pasado ~distante, ni se limita el contexto a pocos elementos)

$$p(\mathbf{x}) = \prod_{t=1}^{|\mathbf{x}|} p(x_t | x_1, ..., x_{t-1})$$

- Varios modelos dependiendo del tipo de red: NNLM (Perceptrón Multicapa), RNNLM (Recurrent Neural Network) y GPT (Transformer).

Modelos de lenguaje causales (NNLM)

- Y. Bengio et al. (2001, 2003)
- Está basado en el perceptrón multicapa.
- En un instante t la red dispone de una representación de un cierto número de palabras previas (w_{t-1},w_{t-2}, …) en la entrada y proporciona como salida una distribución de probabilidad sobre la próxima palabra.
- Se aprende la representación de las palabras y el modelo de predicción simultáneamente.

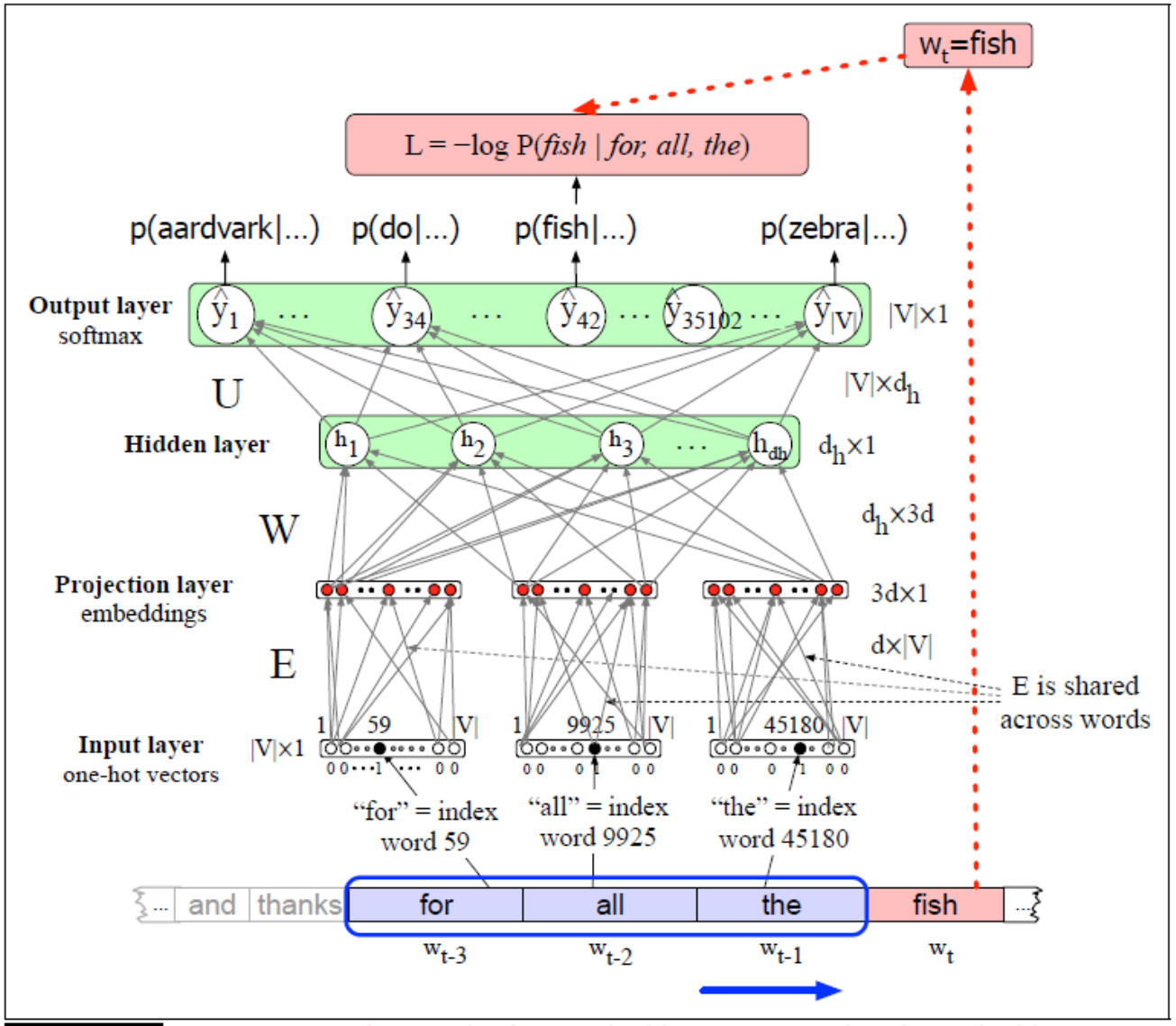
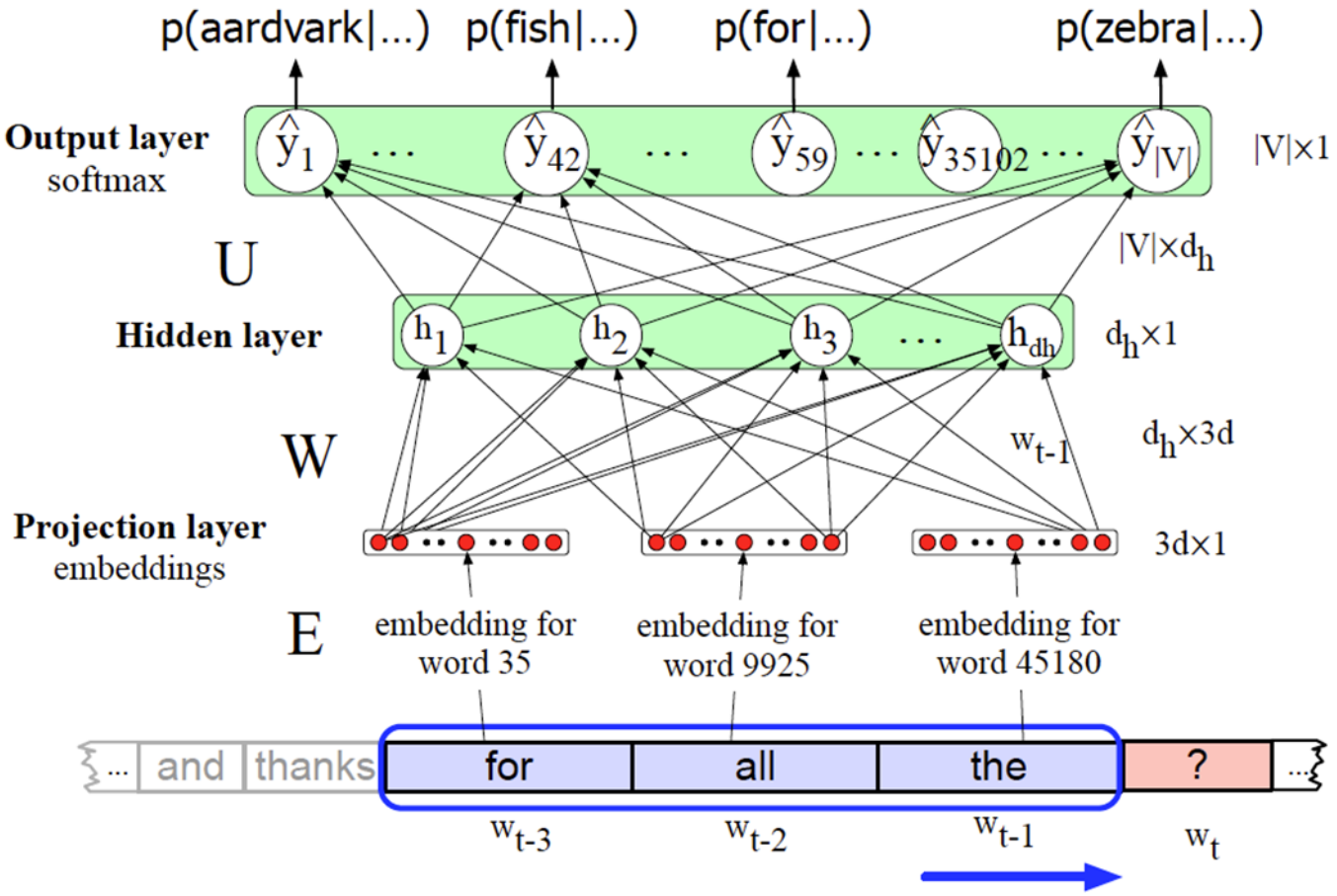


Figure 7.14 Learning all the way back to embeddings. Notice that the embedding matrix *E* is shared among the 3 context words.


p(aardvark|...)
p(fish|...)
p(for|...)
p(zebra|...)
Output layer
softmax
$\hat{y}_1$ ... $\hat{y}_{42}$ ... $\hat{y}_{59}$ ... $\hat{y}_{35102}$ ... $\hat{y}_{|V|}$
|V|×1
U
|V|×d_h
Hidden layer
h_1 h_2 h_3 ... h_{dh}
d_h×1
W
w_{t-1}
d_h×3d
Projection layer
embeddings
3d×1
E
embedding for word 35
embedding for word 9925
embedding for word 45180
... and thanks for all the ? ...
w_{t-3} w_{t-2} w_{t-1} w_t

Modelos de lenguaje causales (NNLM)

- Y. Bengio et al. (2001, 2003)

- Como todos los modelos de lenguaje basados en redes neuronales:
  - Codifica las palabras con vectores de reales (word embeddings).
  - Dimensión de los vectores entre 50 y 1000.

- Ventajas de los word embeddings:
  - Permiten codificar palabras similares (propiedades gramaticales o significados) con vectores cercanos en el espacio vectorial.

Nota sobre los embeddings

- Un embedding, en este contexto, es una representación de una palabra en un espacio vectorial de n dimensiones, donde cada una es situada en dicho espacio vectorial con una representación numérica de valores continuos de n dimensiones –las coordenadas de la palabra–.

- En dicho espacio vectorial los vocablos serán situados más cerca o más lejos de otros en base a lo similares que son los contextos en los que se usan esas palabras, es decir, las que normalmente tengan a su alrededor otras similares, tendrán un contexto más parecido y por tanto se situarán más cerca en espacio que otras que sean usadas en contextos muy distintos, por lo que se puede decir que las palabras cercanas en el espacio son semánticamente similares.

- Una demo para ilustrar: https://projector.tensorflow.org/

Modelos de lenguaje causales (RNNLM)

Recurrent neural network (RNN) based language model, Mikolov (2010)

- En las redes feedforward la historia es representada por un contexto de N-1 palabras, al igual que en el caso de los N-gramas.
- En las redes recurrentes la historia es representada por neuronas con conexiones recurrentes, simulando una longitud de historia ilimitada.
- La redes recurrentes pueden aprender a comprimir la historia completa en un espacio de bajas dimensiones.
- La red recurrente tiene una capa de entrada x, una capa oculta s (también llamada de contexto o estado) y una capa de salida y.
- El vector de entrada x(t) se forma por concatenación del vector w que representa la palabra actual, y la salida de las neuronas en la capa de contexto s en el tiempo t-1.

Modelos de lenguaje causales (RNNLM)

Long Short-Term Memory (LSTM), (Hochreiter, 1997)

- Aunque los modelos RNN podrían aprovechar todos los contextos para la predicción, es un desafío para estos modelos aprender las dependencias a largo plazo en el entrenamiento. La razón es que los gradientes de los parámetros pueden desaparecer o explotar durante el entrenamiento, lo que conduce a un entrenamiento lento o a valores imposibles.

- Para evitar este efecto de escala, el modelo LSTM propone una unidad especial a modo de memoria, que puede almacenar y olvidar información dependiendo de unas puertas (parámetros del modelo). De esta manera, la información puede guardarse entre dos puntos distantes de la secuencia, es decir, los gradientes pueden propagarse por estos nodos para no desvanecerse o explotar tan rápidamente.

Modelos de lenguaje causales (RNNLM)

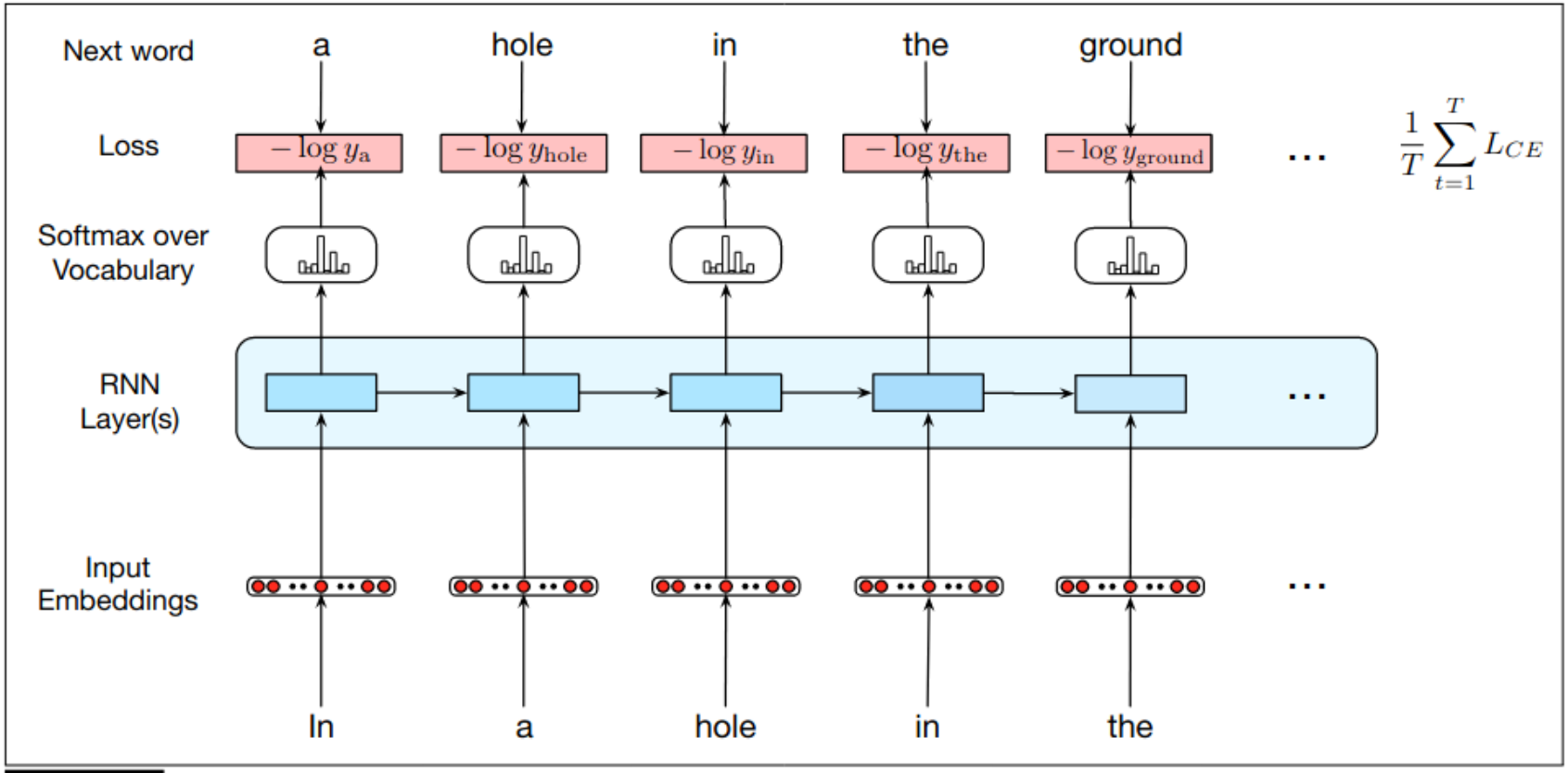


Figure 9.6 Training RNNs as language models.

Modelos de lenguaje causales (RNNLM)

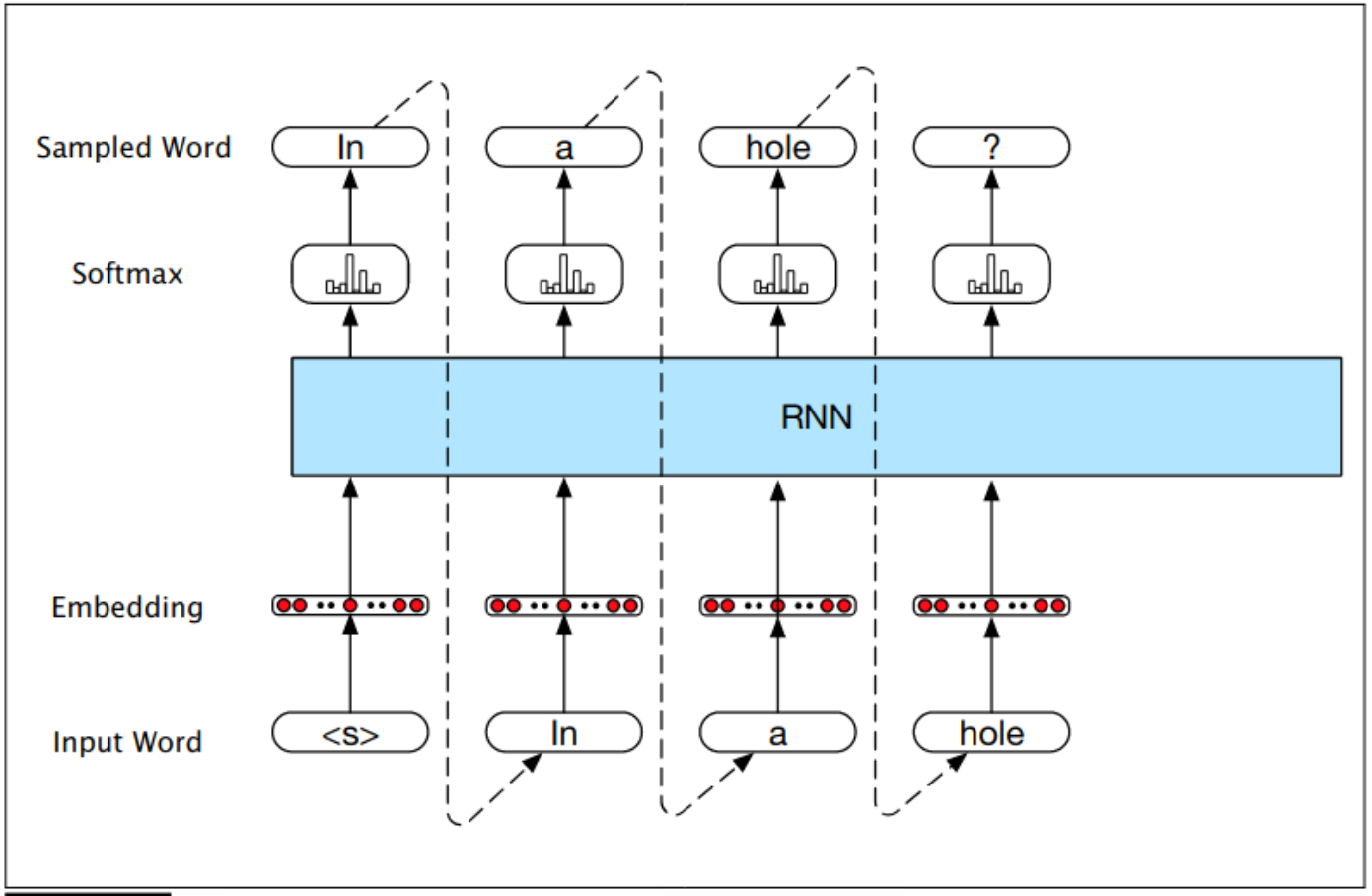


Figure 9.7 Autoregressive generation with an RNN-based neural language model.

Modelos de lenguaje causales (GPT)

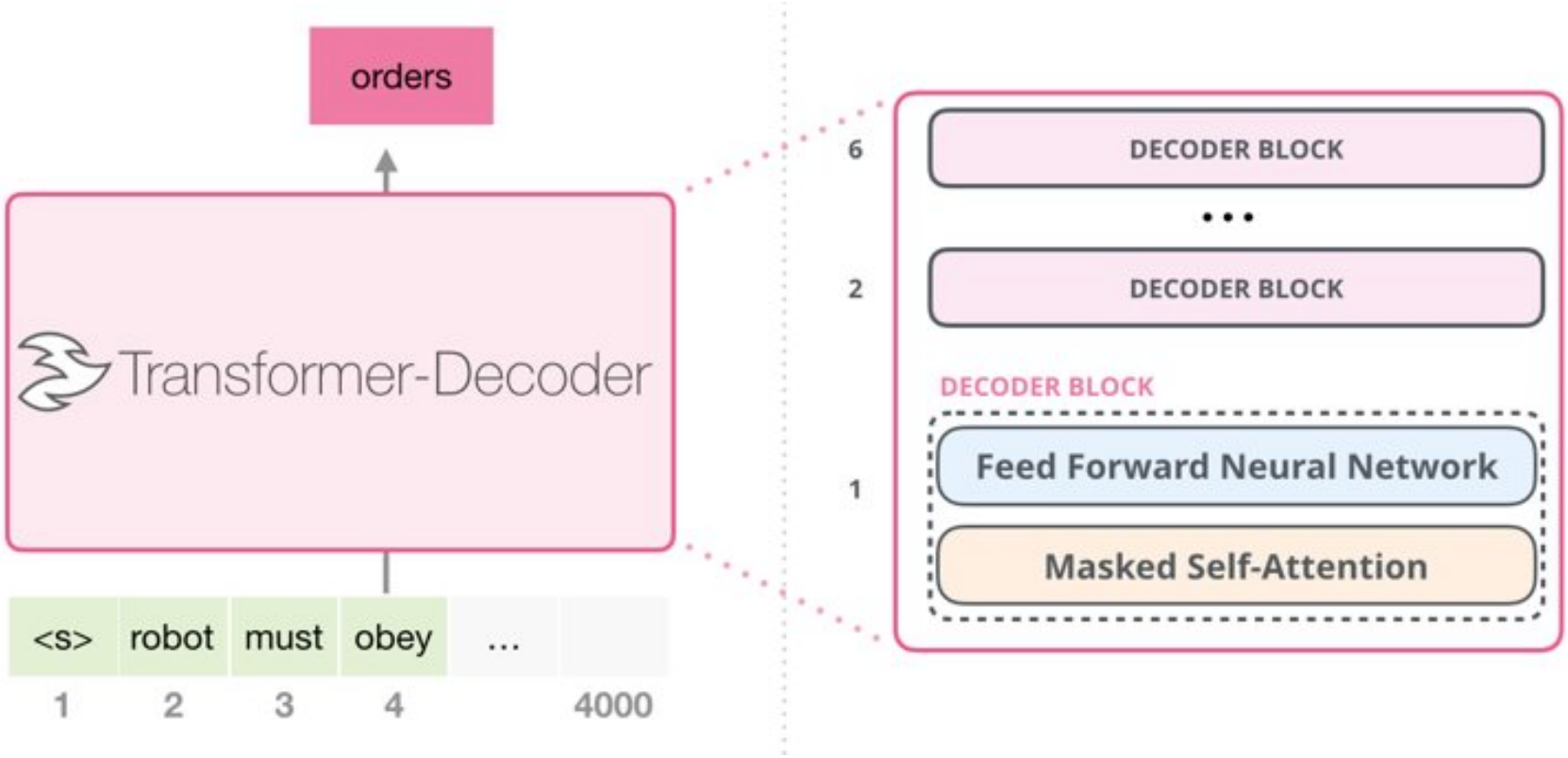
Language Models are Unsupervised Multitask Learners (GPT-2) (Radford et al., 2019)
Language Models are Few-Shot Learners (GPT-3) (Brown et al., 2020)

- Misma idea que RNNLM, sustituyendo RNN por Transformers (Vaswani et al, 2017).

- Los Transformers son la clave de su éxito:
  - No hay secuencialidad (mecanismos de atención), no necesitan backpropagation y son paralelizables, lo que permite definir redes más profundas, manejar contextos más largos y entrenar eficientemente con ingentes cantidades de datos.

- GPT-3 tiene 17 veces más parámetros que GPT-2 (175b) y está entrenado con cientos de miles de millones de palabras (subwords!).

Modelos de lenguaje causales (GPT)



Mucha más información en: https://jalammar.github.io/illustrated-gpt2/

Modelos de lenguaje causales (decoding)

Una vez entrenados, para generar secuencias, típicamente queremos encontrar el sufijo más probable dado un prefijo de longitud t:

$$\mathbf{w}_t^{T*} = \underset{\mathbf{w}_t^T}{\operatorname{argmax}}\ p(\mathbf{w}_t^T|\mathbf{w}_1^t)$$

No es posible encontrarlo (demasiados caminos para explorar). Varias estrategias de decoding (entre otras):

- Greedy search: a cada paso, se coge la palabra que maximiza la probabilidad a posteriori.
- Beam search: mantiene hipótesis en cada paso y elige la que tiene la mayor probabilidad general.
- Nucleus sampling: a cada paso, se determina el mínimo número de palabras cuya probabilidad acumulada supera un umbral , se redistribuye la masa de probabilidad entre estas y se muestrea una palabra. Es no determinista y especialmente útil en open-ended generation (GPT).

Comparación entre N-gramas y modelos de lenguaje causales basados en redes neuronales

- Sea la frase de entrenamiento: “I have to make sure when I get home to feed the cat”.
- Supongamos que en el test aparece: “I forgot when I got home to feed the”.
- Supongamos que en entrenamiento no se ha visto “dog” después de ”feed the”.

El modelo de N-gramas asignará una probabilidad alta a “cat”, sin embargo, asignará a “dog” la probabilidad a través del suavizado (generalmente muy baja).

El modelo de NN con embeddings similares para “cat” y “dog”, aunque en entrenamiento no haya visto “feed the dog”, asignará una probabilidad similar (razonablemente alta) a “cat” y a “dog” tras el prefijo “I forgot when I got home to feed the”.

Los embedding estimados para las palabras “cat” y “dog” son vectores similares, ya que tienen significados y categoría morfosintácticas similares. Esto permite una generalización con estos ML que no es posible con los modelos de N-gramas.

Comparación entre N-gramas y modelos de lenguaje causales basados en redes neuronales

- Consiguen tratar historias de mayor tamaño.
- Generalizan sobre contextos de palabras similares. La codificación consigue que a las palabras que se comportan de forma similar se les asignen códigos similares.
- No se requieren técnicas de suavizado.
- Se consiguen buenas generalizaciones para los eventos no vistos.

Sin embargo, hay un coste asociado al uso de Redes Neuronales como Modelos de Lenguaje:

Los modelos son bastante más lentos de entrenar que los modelos de lenguaje tradicionales, por lo que para muchas tareas los modelos de N-gramas siguen siendo la tecnología adecuada.

Representaciones vectoriales de las palabras (Speech and Language Processing (3rd ed. draft) Dan Jurafsky and James H. Martin)

Tradicionalmente en el procesamiento del lenguaje natural, las palabras se han representado como un elemento de un conjunto (un índice en un diccionario).

Sin embargo, para múltiples aplicaciones, por ejemplo, el cálculo de la similitud semántica, se hace necesario establecer distancias o similitudes entre palabras.

Para ello conviene representar las palabras en un espacio vectorial, de forma que cada palabra tendrá asociado un vector, y por tanto, un punto en ese espacio.

Representaciones vectoriales de las palabras:

- Vectores dispersos (matriz palabra-palabra)
- Vectores densos (embeddings)

Representaciones vectoriales de las palabras

- Ejemplo:

```
A bottle of tesgüino is on the table
Everybody likes tesgüino
Tesgüino makes you drunk
We make tesgüino out of corn.
```

- Por las palabras del contexto, los humanos podemos conjeturar que el significado de ***tesgüino*** es una bebida alcohólica como la cerveza.
- Criterio para la representación vectorial de las palabras (hipótesis distribucional):
  - Dos palabras son similares si aparecen en contextos similares.

Firth (1957):

"You shall know a word by the company it keeps!"

Representaciones vectoriales de las palabras

Vectores dispersos (matriz palabra-palabra)

- Una palabra se representa con un vector en base a las palabras de su contexto: se define un ventana de $\pm$ 7 palabras, por ejemplo.
- Dos palabras son similares si sus vectores de contexto son similares.
- El vector es de dimensión igual a la talla del vocabulario V.
- La matriz palabra-palabra es de talla |V| x |V|.

	aardvark	computer	data	pinch	result	sugar	...
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	

Representaciones vectoriales de las palabras

Vectores dispersos (matriz palabra-palabra)

- Las simples frecuencias de ocurrencia en unos textos de referencia no representan una buena medida para la asociación de palabras.
- Se necesita una medida que represente mejor si una palabra de contexto es particularmente informativa sobre la palabra objetivo:

  Positive Pointwise Mutual Information (PPMI)

Para más información:

Speech and Language Processing (3rd ed. draft)

Dan Jurafsky and James H. Martin

Representaciones vectoriales de las palabras

Vectores densos (embeddings)

Los vectores PPMI son:

- **grandes** (longitudes |V|= 20,000 to 50,000)
- **dispersos** (la mayor parte de los componentes son cero)

Alternativa, estimar vectores que sean:

- **pequeños** (longitudes 200-1000)
- **densos** (la mayor parte de los componentes son distintos de cero)

Representaciones vectoriales de las palabras

Vectores densos (embeddings)

Varios métodos para obtener vectores densos:

- Singular Value Decomposition (SVD)

  un caso particular es el conocido Latent Semantic Analysis (LSA)
- Brown clustering
- Neural Language Models
  - basado en modelos predictivos. Dos tipos: incontextuales (Skip-Gram y CBOW) y contextuales (ELMO, BERT, RoBERTA, AlBERT, ELECTRA, ...).

Para más información:

Speech and Language Processing (3rd ed. draft)

Dan Jurafsky and James H. Martin

Representaciones vectoriales de las palabras

Embeddings obtenidos mediante Neural Language Models

Los modelos de lenguaje sirven como base para extraer conocimiento lingüístico (codificado en los embeddings) transferible a tareas como clasificación de texto o similitud semántica.

Los embeddings se estiman, junto con el resto de parámetros de la red, como parte del proceso de entrenamiento del modelo de lenguaje.

La unidireccionalidad de los modelos de lenguaje causales (NNLM, RNNLM, GPT, …) limita el contexto con el que aprender los embeddings.

Por lo que, típicamente, se usan otros tipos de modelos de lenguaje que consideran contextos bidireccionales. Entre ellos: Skip-Gram, CBOW y Masked Language Modeling (MLM).

Representaciones vectoriales de las palabras

Embeddings incontextuales (o estáticos)

Se dispone de un único embedding para cada palabra del vocabulario, lo que requiere aprender una matriz de embeddings (|V|xd), situada como primera capa de la red, junto con el resto de parámetros de la red.

Una vez entrenado el modelo, se extrae la matriz de embeddings y se descarta el resto. Posteriormente, se usa la matriz para representar las palabras de entrada a otro modelo dedicado a una tarea específica e.g. similitud semántica o clasificación de texto.

Veremos los modelos Skip-gram y CBOW.

Representaciones vectoriales de las palabras

Skip-gram y **CBOW** (Mikolov et al. 2013)

Se entrena una red neuronal para predecir las palabras del contexto.

Dos aproximaciones para entrenar la red:

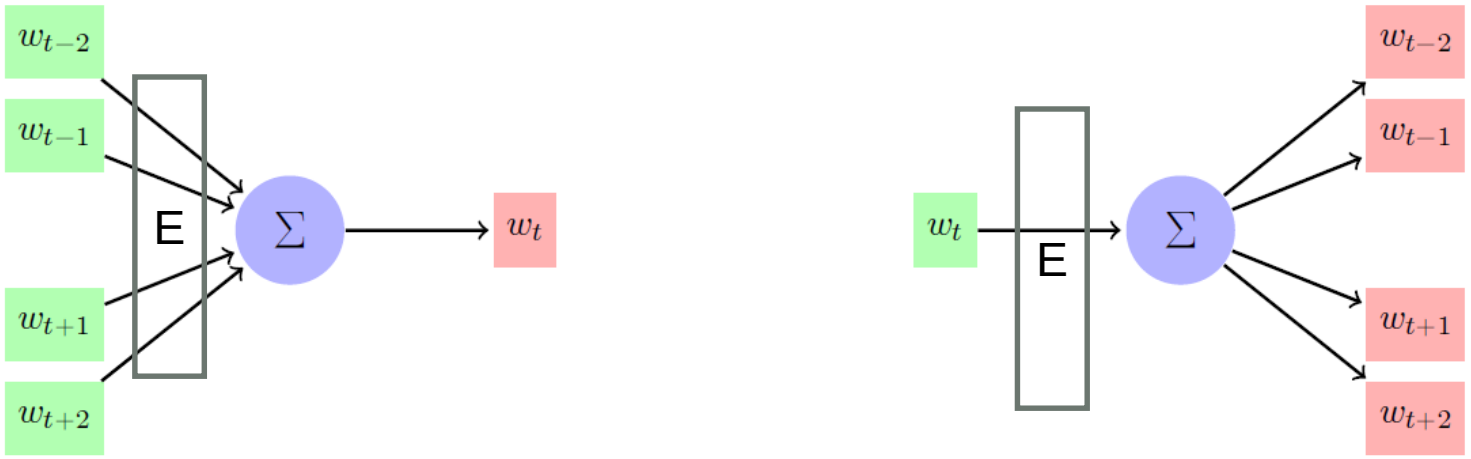
- Continuous Bag of Words (CBOW): la tarea del modelo es predecir la palabra central en una ventana de tamaño h, a partir de las palabras que residen a su alrededor, dentro de la ventana.
- Skip-gram, el modelo debe predecir, a partir de la palabra central, el resto de palabras en una ventana de dimensión h.

Herramientas (*word2vec*, *fastText*, entre otros) y conjuntos de embeddings preentrenados!

Representaciones vectoriales de las palabras

Skip-gram y **CBOW** (Mikolov et al. 2013)

En la figura podemos ver las dos arquitecturas ejemplificadas para un tamaño de ventana de 5 y la palabra en la posición t.



(a) CBOW

(b) Skip-gram

Representaciones vectoriales de las palabras

Skip-gram y **CBOW** (Mikolov et al. 2013)

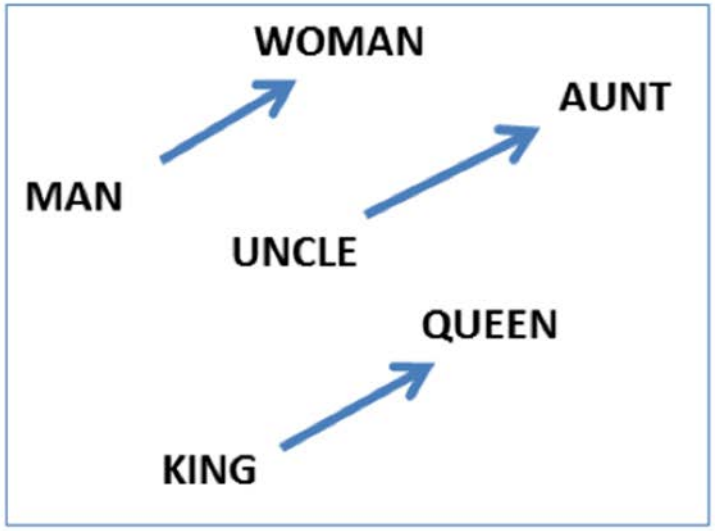
Puesto que los embeddings son la representación interna que obtiene el modelo después de haber sido entrenado, utilizar una arquitectura u otra proporciona prestaciones distintas:

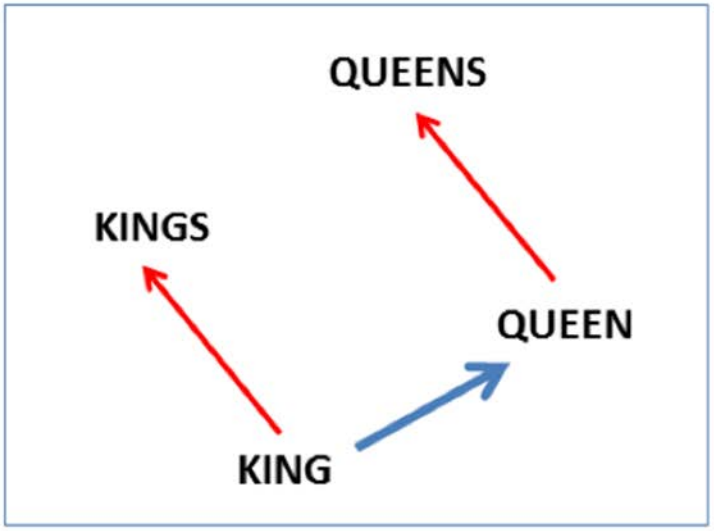
- La arquitectura CBOW es más rápida en el proceso de entrenamiento y representa mejor los términos más frecuentes de un dataset.
- La arquitectura Skip-gram requiere de más tiempo para ser entrenada, pero funciona mejor en datasets más pequeños y consigue representar mejor las palabras menos frecuentes.

Representaciones vectoriales de las palabras

Propiedad de los embeddings: capturan relaciones semánticas

vector(*'king'*) - vector(*'man'*) + vector(*'woman'*) ≈ vector('queen')
vector(*'Paris'*) - vector(*'France'*) + vector(*'Italy'*) ≈ vector('Rome')





Representaciones vectoriales de las palabras

Embeddings contextuales (o dinámicos)

Los embeddings incontextuales colapsan la información de todos los contextos de una palabra en un único vector, lo que puede ser un factor limitante ante fenómenos como la polisemia o aplicaciones donde el contexto sea relevante e.g. detección de ironía.

Lo deseable en estos casos es disponer de un embedding distinto para cada contexto en el que aparece una palabra. *[Notad que esto dificulta el cálculo de la similitud entre palabras, ya que los contextos en dos textos distintos pueden ser muy diferentes para un mismo término].*

Sin embargo, esto no es posible con una única matriz como en el caso de los embeddings incontextuales → Sustituir la matriz por una red neuronal que procese secuencias y usar las salidas de las capas ocultas como embeddings!

Representaciones vectoriales de las palabras

Embeddings contextuales

Dos tipos de modelos de embeddings contextuales:

- **Feature-based**: se usan las salidas de las capas ocultas de la red (congelada) como embeddings para otro modelo específico de la tarea que queramos resolver: ELMO (Peters et al., 2018) y FLAIR (Akbiket al., 2018).
- **Fine-tuning**: la red pre-entrenada se ajusta para la tarea que queramos resolver. Simplemente se añade una capa al final de la red y se reentrena completa para la tarea: BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), AlBERT (Lan et al., 2019), ELECTRA (Clark et al., 2020), etc.

Representaciones vectoriales de las palabras

Embeddings contextuales

Omnipresentes en el estado del arte del NLP (**sota**), especialmente modelos basados en Transformers como BERT, ajustados mediante fine-tuning a una tarea concreta.

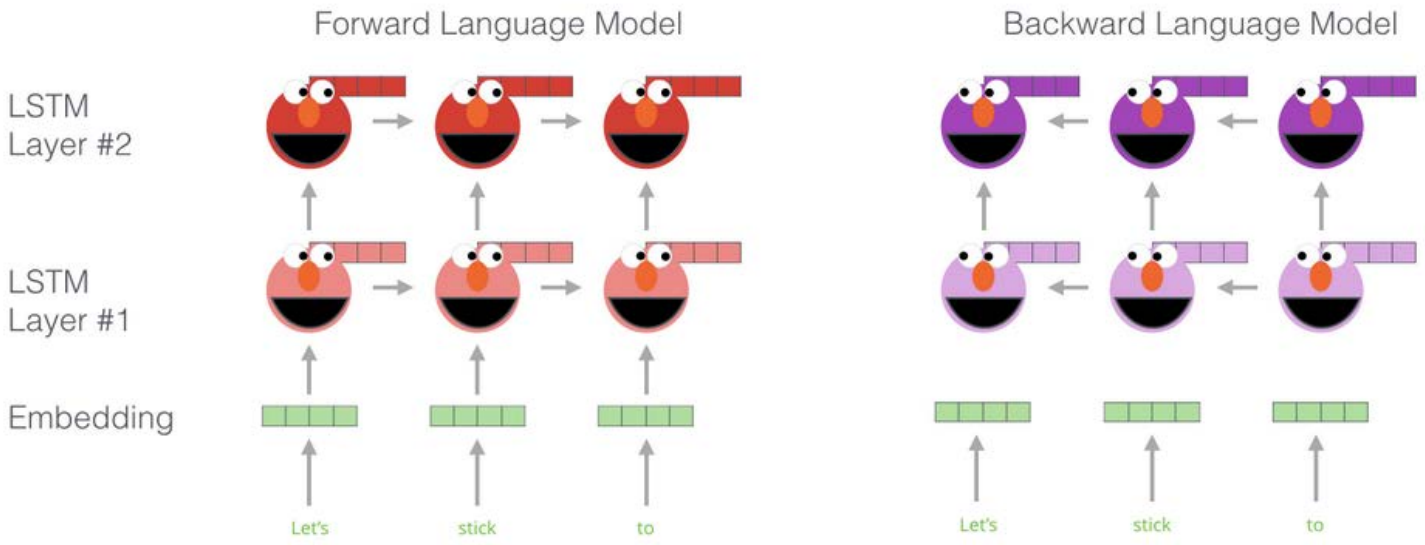
Pueden utilizarse fácilmente con pocas líneas de código mediante herramientas como HuggingFace, AllenNLP o Spacy. Cientos de modelos pre-entrenados y demos!. Pero es casi imprescindible el uso de GPUs.

Veremos ELMO y BERT.

Representaciones vectoriales de las palabras

ELMO (Peters et al., 2018)

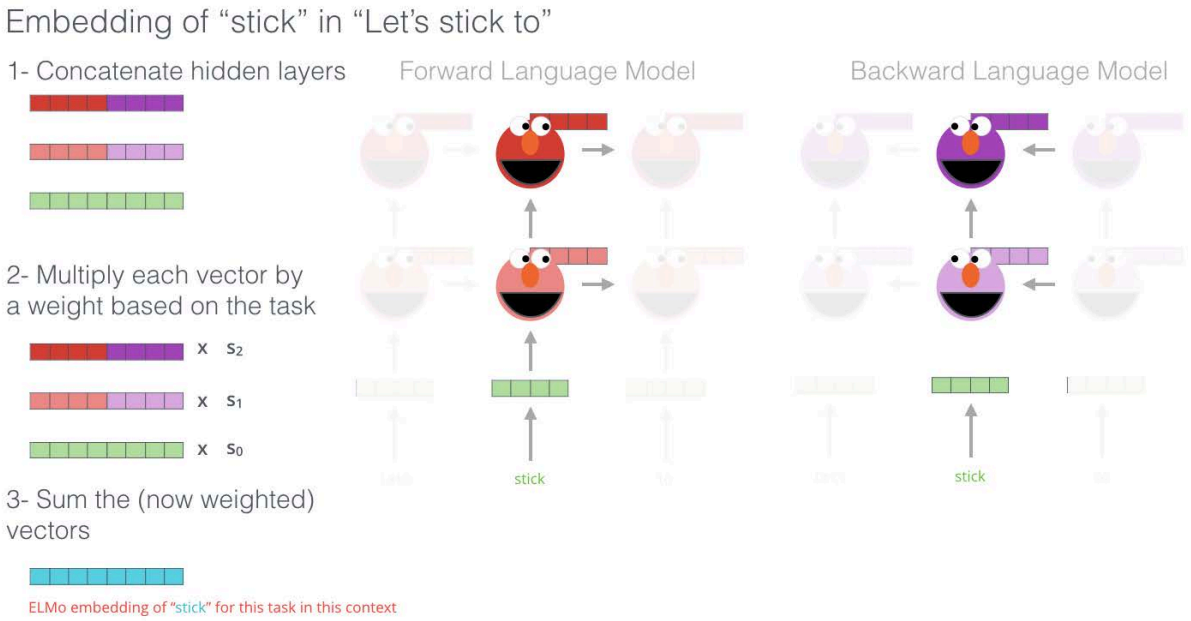
Está basado en LSTMs bidireccionales, se entrena como un modelo de lenguaje causal en ambas direcciones. De izquierda a derecha, en el instante t, se predice el siguiente token y de derecha a izquierda, en el instante t, se predice el token anterior.



Representaciones vectoriales de las palabras

ELMO (Peters et al., 2018)

Una vez entrenado el modelo, el embedding de la palabra en el instante t de la secuencia viene dado por las salidas de las capas ocultas e.g. de la última capa, una suma ponderada de todas las capas, etc.



Representaciones vectoriales de las palabras

BERT (Devlin et al., 2018)

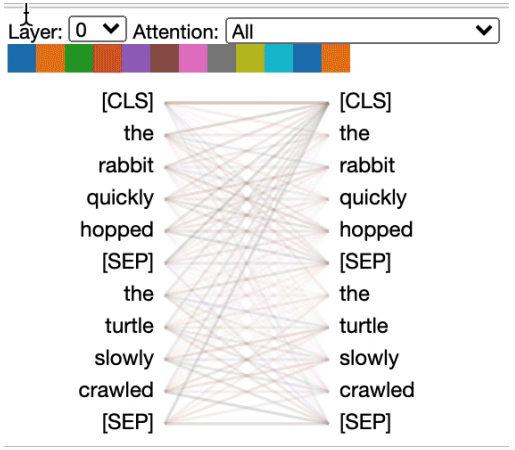
En ELMO, la bidireccionalidad se consigue combinando los procesos de izquierda a derecha y de derecha a izquierda. Una manera más natural, utilizada por BERT, es considerar cloze tasks/fill the gaps como objetivo de entrenamiento. Esto se conoce como Masked Language Modeling (MLM).

Master of Science in Information Technology (MSc in IT): Our programme will develop your [ ] knowledge of Computer Science and your problem-solving and [ ] skills, while enabling you to achieve the [ ] qualification for the IT professional. The programme structure is extremely [ ], enabling you to personalise your MSc through a wide range of electives.

Representaciones vectoriales de las palabras

BERT (Devlin et al., 2018)

Además de redefinir el objetivo de entrenamiento, BERT está basado en Transformers, que, a diferencia de las RNNs, no presentan secuencialidad (todas las palabras se relacionan con el resto en un único paso mediante los mecanismos de atención).

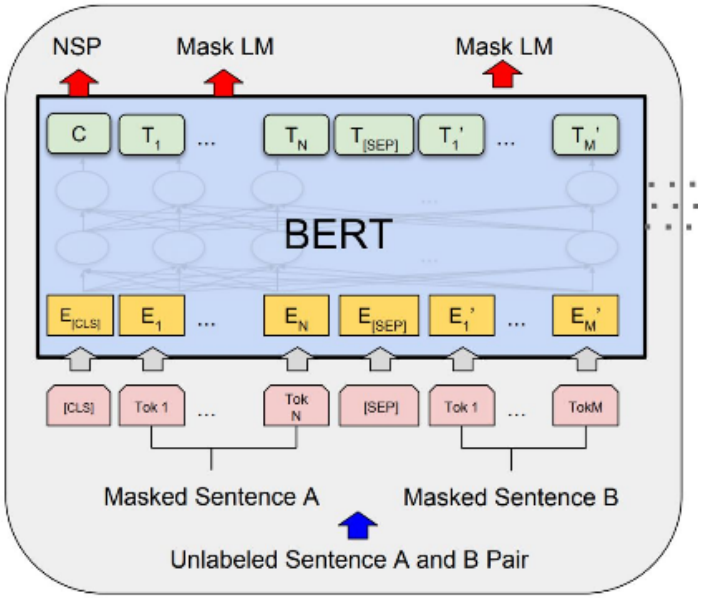


Esto permite modelar mejor la bidireccionalidad y aporta mejoras prácticas (como en GPT en comparación a RNNLM: paralelización y entrenamiento con ingentes cantidades de datos)

Representaciones vectoriales de las palabras

BERT (Devlin et al., 2018)

Al entrenamiento del modelo con MLM y NSP, típicamente usando grandes colecciones de texto no etiquetado (en BERT, Wikipedia+BookCorpus ~3.3B de palabras del inglés), se le conoce como “pre-training”.



Pre-training

Representaciones vectoriales de las palabras

BERT (Devlin et al., 2018)

Una vez entrenado el modelo, se pueden extraer las representaciones de palabras a partir de las capas ocultas de la red, como en ELMO.

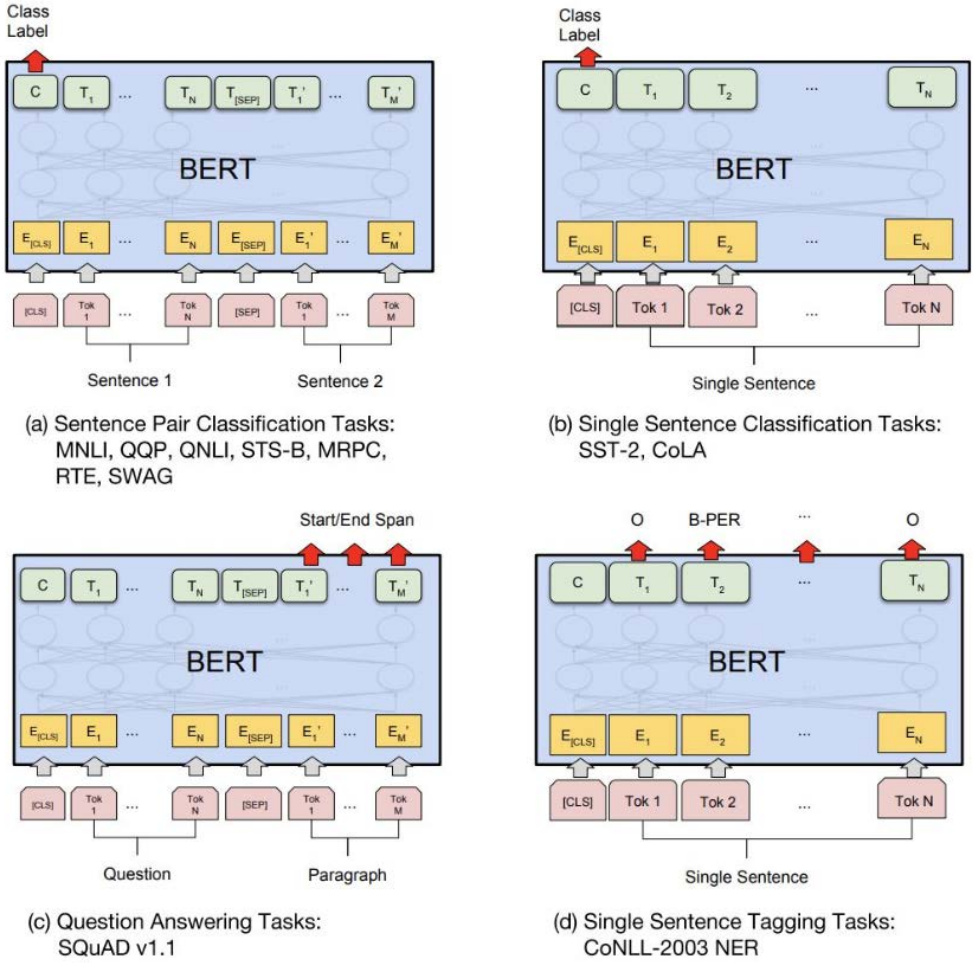
Si se va a abordar un problema del que se dispone de un corpus etiquetado, la receta estándar que mejores resultados reporta consiste en hacer finetuning i.e. añadir capas adicionales, inicializadas aleatoriamente, justo después de la última capa (descartando la capa softmax del pre-entrenamiento) y reentrenar con el corpus etiquetado.

De esta manera se puede ajustar el modelo de representación, al mismo tiempo que se aprende el modelo específico para la tarea.

Representaciones vectoriales de las palabras

BERT (Devlin et al., 2018)

Ejemplos de finetuning con tareas de clasificación y etiquetado de secuencias (single/multiple input y single/multiple output).



Representaciones vectoriales del texto

Los embeddings simbolizan una palabra (o token), no obstante para caracterizar un texto (frases, párrafos, documentos, …) mediante estos, debemos combinar los embeddings de cada una de las palabras que forma el texto y unirlos en un solo vector (composicionalidad). Se desea que textos semánticamente similares estén cercanos en el espacio de representación.

Normalmente se utiliza el promedio de la suma de los embeddings de las palabras que forman el texto, de modo que tendremos «la semántica central» del tema que trata el texto.

El problema de este tipo de representaciones es que a medida que el texto se hace más grande, el «embedding del texto» se alejará más del tema central que podría caracterizarlo.

Por tanto, aunque la representación de las palabras adquiera algo de contexto durante el entrenamiento, el contexto dentro de un texto específico puede perderse con esta técnica de representación.

Representaciones vectoriales del texto

Varias alternativas para generar representaciones vectoriales del texto:

- Promedio de word embeddings: especialmente útil cuando se manejan textos cortos e.g. IR o clasificación con una colección de tweets.
- A partir de los componentes de algunos modelos de lenguaje neuronales vistos en esta sesión e.g. del token [CLS] en modelos BERT-like.
- Modelos específicos para aprender embeddings de texto: Doc2Vec (Le et al., 2014), FastSent (Hill et al., 2016), InferSent (Conneau et al., 2017), QuickThought (Logeswaran et al., 2018), SentenceBERT (Reimers et al., 2019), …

Representaciones vectoriales del texto

SentenceBERT (Reimers et al., 2019)

Imaginad que trabajamos en IR, disponemos de un modelo BERT ajustado para una tarea supervisada de similitud semántica (dadas dos frases como entrada, el modelo devuelve un valor numérico entre [0, 1], que representa la similitud) y queremos encontrar el par de frases más similares en una colección de n=10000 documentos. Esto requiere n*(n-1)/2 inferencias (~50 millones) con un modelo muy costoso! (65 horas en sus experimentos).

Podemos pensar en utilizar el embedding del token [CLS] o promediar los embeddings de las palabras (dando como entrada a BERT una única frase) y luego calcular similitudes coseno, pero esto generalmente conduce a peores resultados que simplemente promediar embeddings incontextuales.

Representaciones vectoriales del texto

SentenceBERT (Reimers et al., 2019)

La principal idea de SentenceBERT es derivar embeddings de frases, de manera que dos frases semánticamente similares estén cerca en el espacio de representación. Por tanto, usando métricas de similitud como la similitud coseno, podemos encontrar frases similares.

Reduce el coste temporal de buscar el par más similar en una colección de 10000 frases de 65 horas (BERT) a ~5 segundos.

Mejora hasta 50 puntos, en el benchmark STS, en comparación a otras estrategias como usar el token [CLS] de BERT (mayor calidad de los embeddings de frases)

Representaciones vectoriales del texto

SentenceBERT (Reimers et al., 2019)

Se entrena mediante grandes corpus supervisados de NLI (textual entailment), con la idea de que si una hipótesis es cierta dada una premisa, las representaciones de ambas (u y v en la figura) deben ser similares.

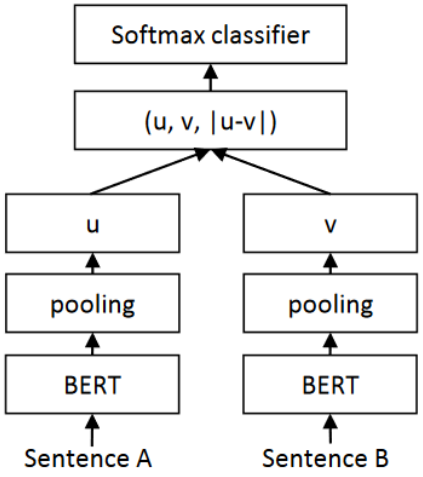


Figure 1: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure).

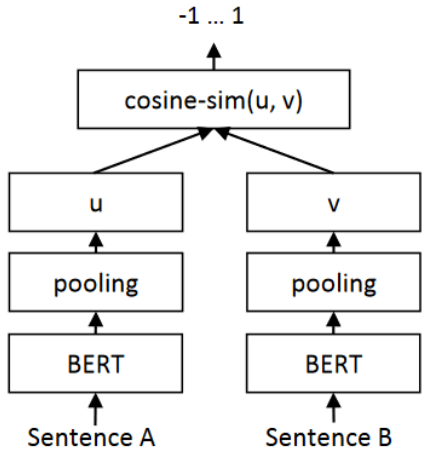


Figure 2: SBERT architecture at inference, for example, to compute similarity scores. This architecture is also used with the regression objective function.