PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

FACULTAD DE LETRAS

Introducción a la lingüística computacional

César Antonio Aguilar

Facultad de Lenguas y Letras

24/10/2017

Cesar.Aguilar72@gmail.com

¿Probabilidades y sintaxis? (1)

En esta clase vamos a abordar la relación que mantiene, dentro de la lingüística computacional, el estudio de la sintaxis con el uso de métodos gramaticales.

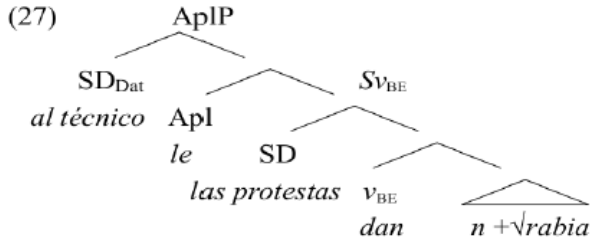
Durante los primeros años de desarrollo del PLN, estas cuestiones eran vistas como dos metodologías distintas entre sí:

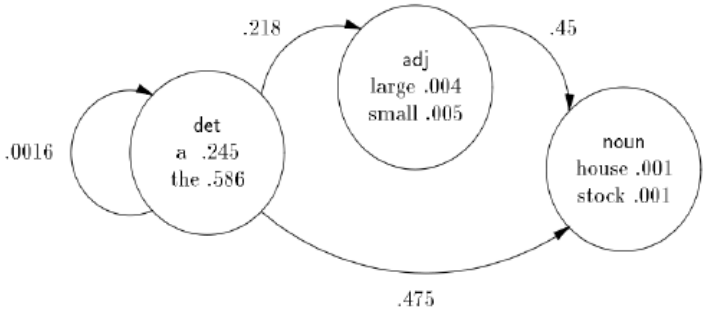
1. Métodos simbólicos: aquellos que basan su análisis en la relación que se da reglas y parámetros de aplicación, conforme a una hipótesis.

2. Métodos estocásticos: aquellos que basan su análisis el uso de técnicas estadísticas para determinar frecuencias de uso y realizar predicciones.

(26) *Al técnico le dan rabia las protestas*

(27)





¿Probabilidades y sintaxis? (2)

De acuerdo con Chris Martin (2007), los verbos *to regard*, *to consider* y *to think* tienen dos comportamientos sintácticos diferentes, debido al tipo de complementos que requieren para configurar una oración.

regard: __ NP[acc] *as* {NP, AdjP}

consider: __ NP[acc] {AdjP, NP, VP[inf]}

think: { __ CP[that], __ NP[acc] NP }

En el caso de *to regard* y *to consider*, se supone que el primero necesita asociarse con la preposición *as* para configurar una predicación secundara; en tanto que el segundo no requiere tal preposición (de hecho, parece ser que si se ligara *as*, se tendría una construcción agramtical).

¿Probabilidades y sintaxis? (3)

Podemos ver mejor la distinción entre ambos verbos a partir de los siguientes ejemplos:

a) We regard Kim as an acceptable candidate

b) * We regard Kim to be an acceptable candidate

c) We consider Kim to be an acceptable candidate

d) *We consider Kim as an acceptable candidate

Estas diferencias fueron planteadas por Carl Pollard e Ivan Sag en 1994, en un trabajo sobre sintaxis basado en HPSG.

¿Probabilidades y sintaxis? (4)

Sin embargo, parece que la gente del *New York Times* no sigue esta regla del todo. Por ejemplo, el verbo *to regard* no siempre aparece ligado a *as*:

a) As 70 to 80 percent of the cost of blood tests, like prescriptions, is paid for by the state, neither physicians nor patients regard expense to be a consideration.

¿Probabilidades y sintaxis? (5)

Y en contraparte, *to consider* puede asociarse sin problemas a la preposición *as*, según la gramática del NYT:

a) The boys consider her as family and she participates in everything we do.

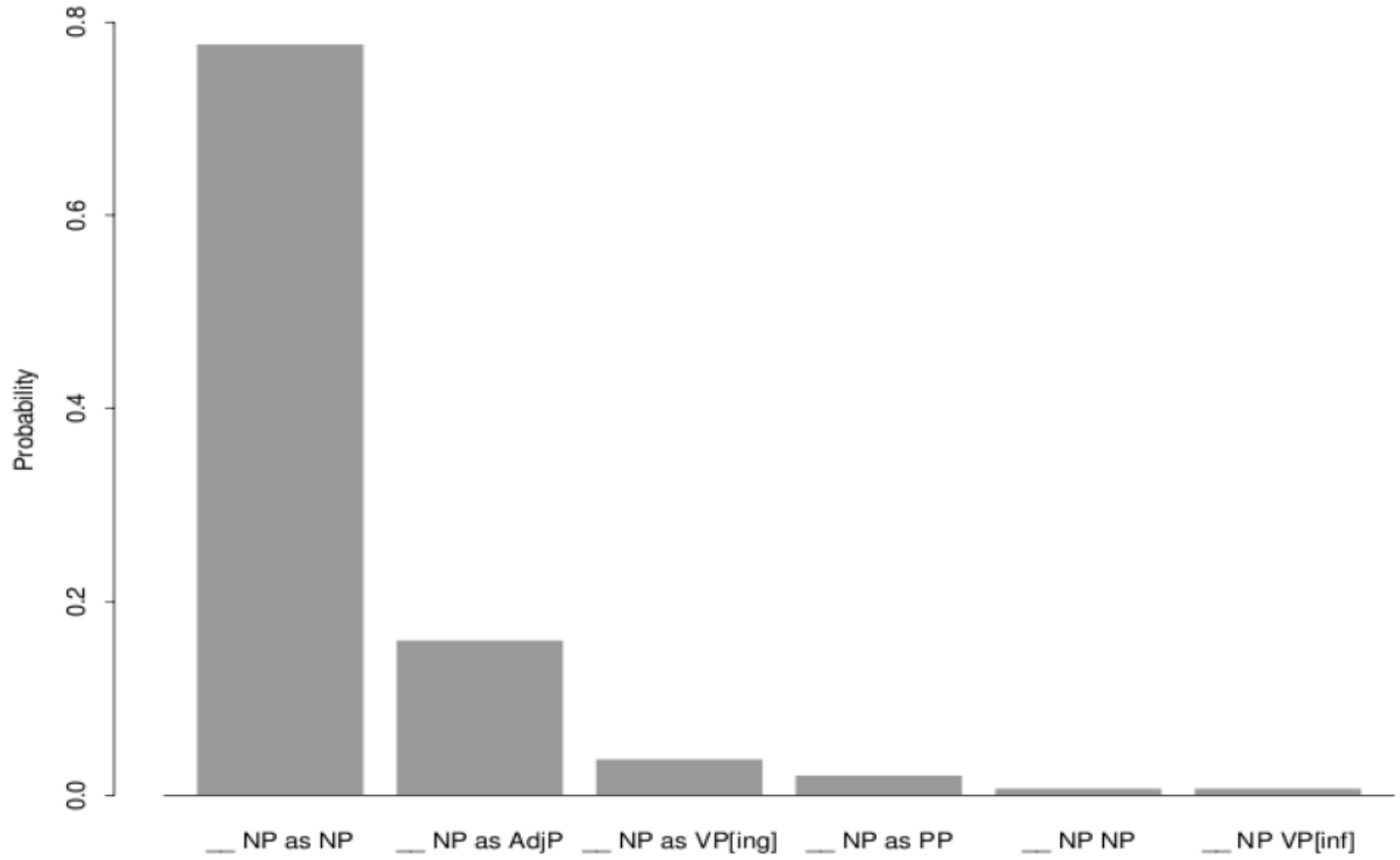
b) Greenspan said, “I don’t consider it as something that gives me great concern.”

c) “We consider that as part of the job,” Keep said.

d) Although the Raiders missed the playoffs for the second time in the past three seasons, he said he considers them as having championship potential.

¿Probabilidades y sintaxis? (6)

Lo más curioso: si argumentamos que los “malos ejemplos” son casos aislados, los conteos nos dan una sorpresa:



P&S (1994): ✓ ✓ ✓ ✓ ✗ ✗

¿Probabilidades y sintaxis? (7)

Manning propone la siguiente reflexión tras revisar estos casos:

prepositional dative structure:	… *gave* [*toys*] [*to the children*]	V NP PP
double object structure:	… *gave* [*the children*] [*toys*]	V NP NP
dative PP:	… *gave* [*toys*] [***to the children***]	V NP **PP**
dative NP:	… *gave* [***the children***] [*toys*]	V **NP** NP
theme:	… *gave* [***toys***] [*to the children*]	V **NP** PP
	… *gave* [*the children*] [***toys***]	V NP **NP**
recipient:	… *gave* [*toys*] [***to the children***]	V NP **PP**
	… *gave* [***the children***] [*toys*]	V **NP** NP

Como lingüistas formales consideramos estos ejemplos como posibles anomalías explicables en términos de variantes dialectales, sociales, culturales, etc.; pero insistimos en que no afectan el sistema gramatical “radicalmente”.

Como lingüistas de corpus, nos sentimos tentados a decir que este es un claro reflejo de cambios gramaticales, ya sea en forma de alternancias, ya sea incluso como nuevas estructuras a punto de consolidarse.

	Coefficient	Odds Ratio PP	95% C.I.
inanimacy of recipient	2.54	12.67	5.56–28.87
nonpronominality of recipient	1.17	3.22	1.70–6.09
nongivenness of recipient	0.99	2.69	1.37–5.3
transfer semantic class	0.96	2.61	1.44–4.69
indefiniteness of recipient	0.85	2.35	1.25–4.43
plural number of theme	0.50	1.65	1.05–2.59
person of recipient	0.48	1.62	1.06–2.46
nongivenness of theme	–1.05	0.35	0.19–0.63
structural parallelism in dialogue	–1.13	0.32	0.22–0.47
nonpronominality of theme	–1.18	0.31	0.19–0.50
length difference (log scale)	–1.21	0.3	0.22–0.4
communication semantic class	–1.34	0.26	0.13–0.55
indefiniteness of theme	–1.37	0.25	0.15–0.44

¿Probabilidades y sintaxis? (8)

Veamos otro ejemplo: en gramática, un problema sumamente interesante es la resolución de ambigüedades como la siguiente:

Supongamos que estamos en un aeropuerto de Estados Unidos, y tenemos que reservar un boleto de avión por una aerolínea llamada TWA.

Hasta allí, todo es normal, salvo que esta reservación la hacemos con uno de esos novedosas máquinas electrónicas de diálogo. En algún momento, le hacemos la siguiente petición a la máquina:

Can you book TWA flights?

¿Probabilidades y sintaxis? (9)

¿Cuál es el problema con esta oración? Que por lo menos tiene dos sentidos:

1. **Can you book TWA flights?** = *Can you book* ***on behalf of*** *TWA flights?* ("¿Puedes reservarme un vuelo **a nombre de** TWA?").

2. **Can you book TWA flights?** = *Can you book flights* ***run by*** *TWA?* ("¿Puedes reservarme un vuelo **operado por** TWA?").

No es del todo "fácil" resolver este problema de ambigüedad. Para hacerlo, la máquina puede seguir dos caminos:

1.Digamos que no es muy lista, entonces le pedirá más datos a su cliente para saber qué quiere hacer.
2.Si es muy lista, a la mejor puede inferir lo que nosotros queremos, y plantearlo como una primera opción (y si no es eso, entonces pasa a una segunda opción).

¿Probabilidades y sintaxis? (10)

De momento, no es necesario profundizar en los tecnicismos. Jurafsky y Martin, en su manual sobre PLN, proponen que la máquina puede deducir los dos sentidos de la oración, haciendo un análisis sintáctico con una gramática libre de contexto (GLC), a la cual se le implementa además un método probabilístico condicional.

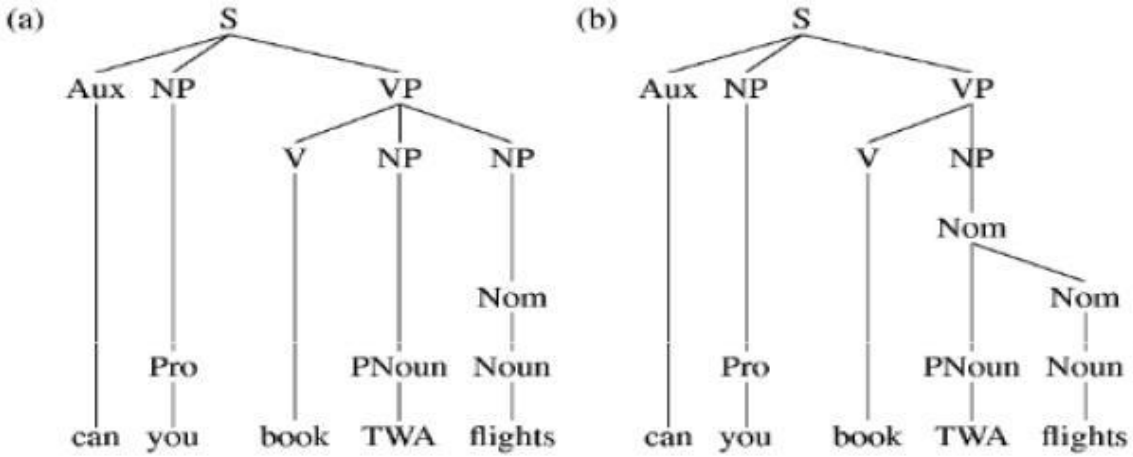
$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow that$ [.05] | the [.80] | a	[.15]
$S \rightarrow Aux\ NP\ VP$	[.15]	$Noun \rightarrow book$	[.10]
$S \rightarrow VP$	[.05]	$Noun \rightarrow flights$	[.50]
$NP \rightarrow Det\ Nom$	[.20]	$Noun \rightarrow meal$	[.40]
$NP \rightarrow Proper\text{-}Noun$	[.35]	$Verb \rightarrow book$	[.30]
$NP \rightarrow Nom$	[.05]	$Verb \rightarrow include$	[.30]
$NP \rightarrow Pronoun$	[.40]	$Verb \rightarrow want$	[.40]
$Nom \rightarrow Noun$	[.75]	$Aux \rightarrow can$	[.40]
$Nom \rightarrow Noun\ Nom$	[.20]	$Aux \rightarrow does$	[.30]
$Nom \rightarrow Proper\text{-}Noun\ Nom$	[.05]	$Aux \rightarrow do$	[.30]
$VP \rightarrow Verb$	[.55]	$Proper\text{-}Noun \rightarrow TWA$	[.40]
$VP \rightarrow Verb\ NP$	[.40]	$Proper\text{-}Noun \rightarrow Denver$	[.40]
$VP \rightarrow Verb\ NP\ NP$	[.05]	$Pronoun \rightarrow you$ [.40] | I [.60]	

¿Qué queremos hacer con esto? Determinar cuál de los dos sentidos es más probable, analizando la constitución sintáctica de estas dos oraciones.

¿Probabilidades y sintaxis? (11)

Usando los dos métodos descritos, obtenemos los siguientes árboles sintácticos:

Pregunta: ¿Cuál es la diferencia entre estos dos árboles? ¿La pueden explicar con sus propias palabras?



Rules			P
S	→	Aux NP VP	.15
NP	→	Pro	.40
VP	→	V NP NP	.05
NP	→	Nom	.05
NP	→	PNoun	.35
Nom	→	Noun	.75
Aux	→	Can	.40
NP	→	Pro	.40
Pro	→	you	.40
Verb	→	book	.30
PNoun	→	TWA	.40
Noun	→	flights	.50

Rules			P
S	→	Aux NP VP	.15
NP	→	Pro	.40
VP	→	V NP	.40
NP	→	Nom	.05
Nom	→	PNoun Nom	.05
Nom	→	Noun	.75
Aux	→	Can	.40
NP	→	Pro	.40
Pro	→	you	.40
Verb	→	book	.30
Pnoun	→	TWA	.40
Noun	→	flights	.50

¿Probabilidades y sintaxis? (12)

¿Qué tipo de análisis probabilístico hacemos? Uno que nos indique que la probabilidad de que nuestro análisis genere un árbol pertinente (esto es, un **parser**), se defina como el producto de las probabilidades de todas las reglas r, las cuales han sido empleadas para expandir cada nodo de nuestro árbol:

$$P(T,S) = \prod_{n \in T} p(r(n))$$

O para decirlo de otro modo: "dada la probabilidad de generar un árbol sintáctico T, éste tiene una correspondencia con una oración S, tomando en cuenta una regla determinada".

$$P(T,S) = P(T)P(S|T) = P(T)$$

¿Probabilidades y sintaxis? (13)

Por fortuna, estos cálculos los hace nuestra máquina. Aplicando las fórmulas anteriores, obtenemos:

$$
\begin{aligned}
P(T_l) &= .15 * .40 * .05 * .05 * .35 * .75 * .40 * .40 * .40 \\
&\quad * .30 * .40 * .50 \\
&= 1.5 \times 10^{-6}
\end{aligned}
$$

Can you book on behalf of TWA flights?

$$
\begin{aligned}
P(T_r) &= .15 * .40 * .40 * .05 * .05 * .75 * .40 * .40 * .40 \\
&\quad * .30 * .40 * .50 \\
&= 1.7 \times 10^{-6}
\end{aligned}
$$

Can you book flights run by TWA?

Pregunta: ¿Qué sentido fue el ganador?

Nota importante: el Sentido 2 de la oración no queda descarto, lo que pasará en todo caso es que la máquina será más precisa en su inferencia y nos preguntará algo como: *Do you want to reserve a flight **running by** TWA?*, suponiendo que el verbo *to reserve* sea más fácil de entender que *to book*.

Gramáticas probabilísticas en NLTK (1)

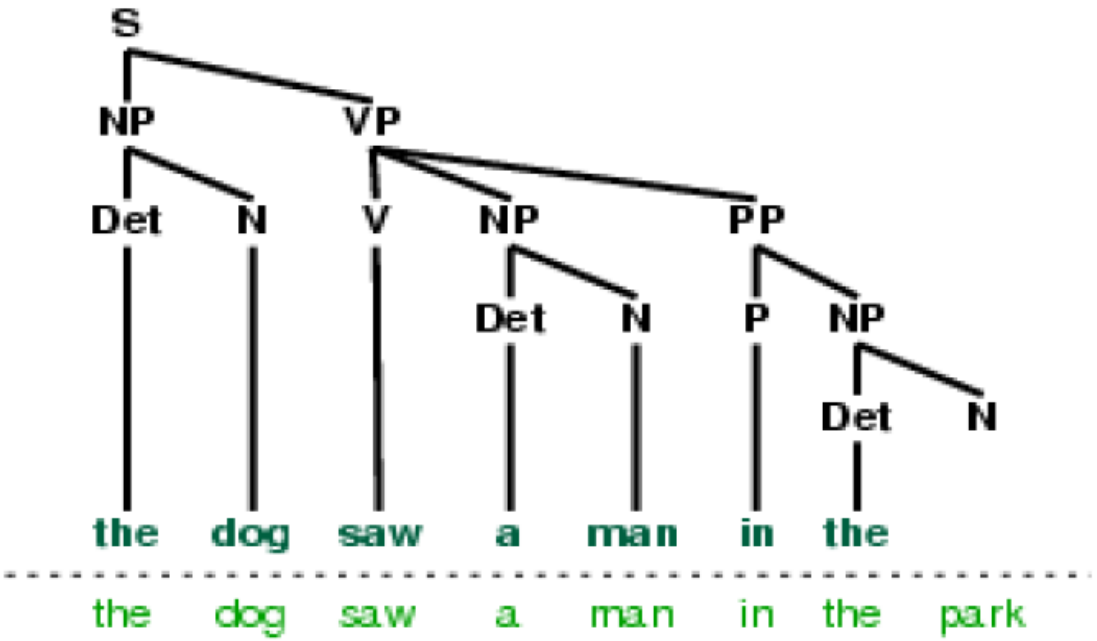
En NLTK podemos modelar gramáticas probabilísticas (o **Probabilistic Context-Free Grammar, PCFG)** usando algunos comandos. Veamos:

```
>>> import nltk. re
>>> grammar1 = nltk.parse_cfg("""
S -> NP VP
VP -> V NP | V NP PP
PP -> P NP
V -> "saw" | "ate" | "walked"
NP -> "John" | "Mary" | "Bob" | Det N | Det N PP
Det -> "a" | "an" | "the" | "my"
N -> "man" | "dog" | "cat" | "telescope" | "park"
P -> "in" | "on" | "by" | "with"
""")
>>> sent = "Mary saw Bob".split()
>>> rd_parser = nltk.RecursiveDescentParser(grammar1)
>>> for tree in rd_parser.nbest_parse(sent):
print tree
```

Gramáticas probabilísticas en NLTK (2)

¿Cómo se pasa de una CFG a una PCGF en NLTK? Podemos decir que es algo “natural”, si tomamos en cuenta cómo se va dando el recorrido en un árbol. Veamos una demostración:

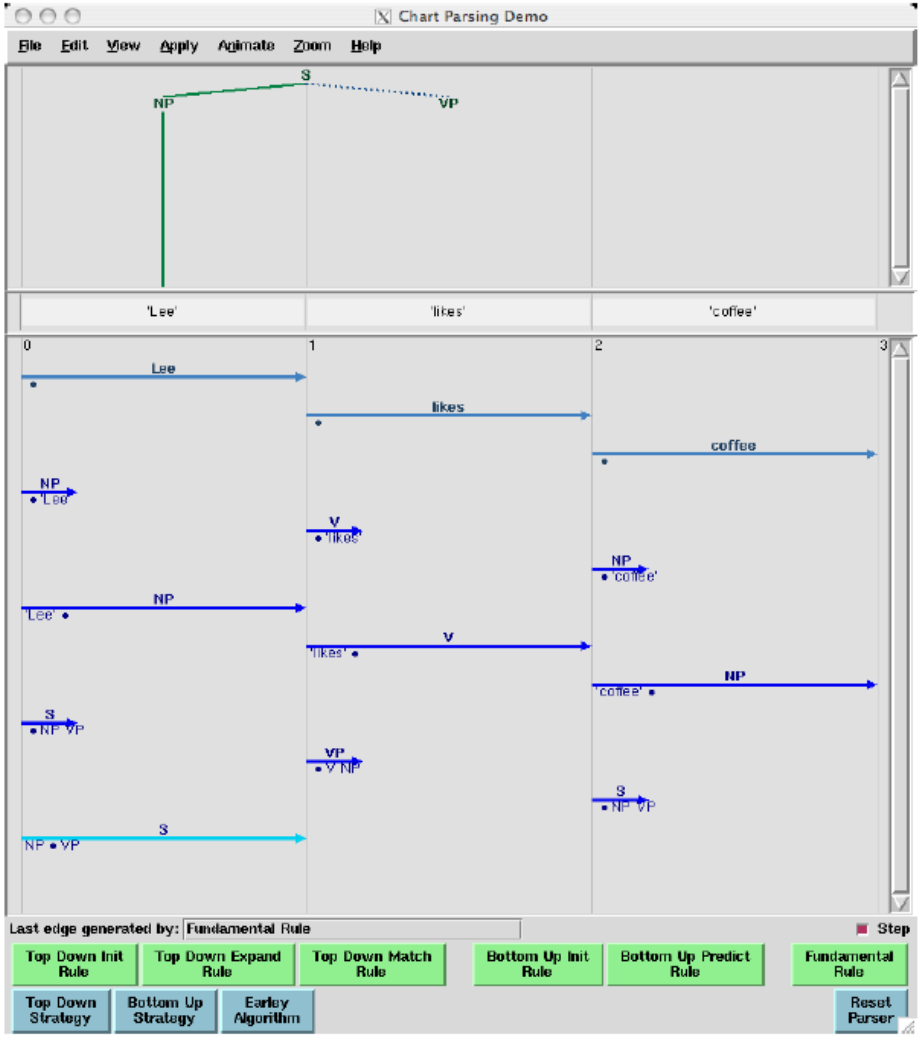
```
>>> import nltk
>>> nltk.app.rdparser()
```



Gramáticas probabilísticas en NLTK (3)

Como podrán notar, esta demo evalúa y genera cuál es el mejor recorrido para derivar un no-terminal a un terminal. El truco es que sigue justo una PCFG tal como las del ejemplo de Jurafsky. Otra forma de ver este recorrido es:

```
>>> import nltk
>>> nltk.app.chartparser()
```



Gramáticas probabilísticas en NLTK (4)

En concreto, NLTK hce uso de de esta clase de gramáticas para generar árboles que, probabilísticamente, sean las mejores opciones para describir estructuralmente una oración.

```
import nltk
Grammar01 = nltk.PCFG.fromstring ("""
S -> NP VP [1.0]
VP -> TV NP [0.4]
VP -> IV [0.3]
VP -> DatV NP NP [0.3]
TV -> 'saw' [1.0]
IV -> 'ate' [1.0]
DatV -> 'gave' [1.0]
NP -> 'telescopes' [0.8]
NP -> 'Jack' [0.2]
""")
print (Grammar01)
```

Gramáticas probabilísticas en NLTK (5)

De hecho, se puede implementar un algoritmo como el de Viterbi para evaluar cuál es la mejor descripción posible:

```
viterbi_parser = nltk.ViterbiParser(Grammar01)
for tree in viterbi_parser.parse(['Jack', 'saw', 'telescopes’]):
    print(tree)
```

Resultado:

```
(S (NP Jack) (VP (TV saw) (NP telescopes))) (p=0.064)
```

Gramáticas probabilísticas en NLTK (6)

¿Qué es lo que estamos haciendo? Básicamente, analizar frases y oraciones asignando un conjunto de probabilidades a cada uno de los componentes sintácticos que conforman una oración, de una forma muy similar al modo en como se pueden identificar secuencias de *n*-gramas:

Span	Node	Tree	Prob
[0:1]	NP	(NP I)	0.15
[6:7]	NP	(NN telescope)	0.5
[5:7]	NP	(NP the telescope)	0.2
[4:7]	PP	(PP with (NP the telescope))	0.122
[0:4]	S	(S (NP I) (VP saw (NP the man)))	0.01365
[0:7]	S	(S (NP I) (VP saw (NP (NP the man) (PP with (NP the telescope)))))	0.0004163250

1. "saw the man, who has the telescope":

   (VP saw

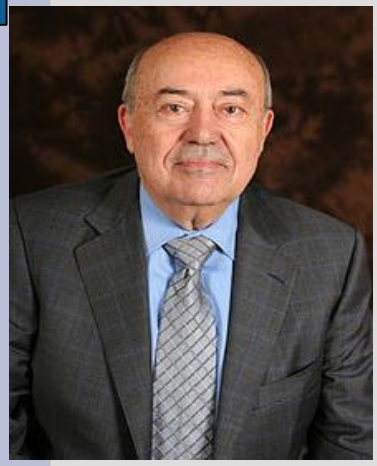
   (NP (NP John)
   (PP with (NP the telescope))))

2. "used the telescope to see the man":

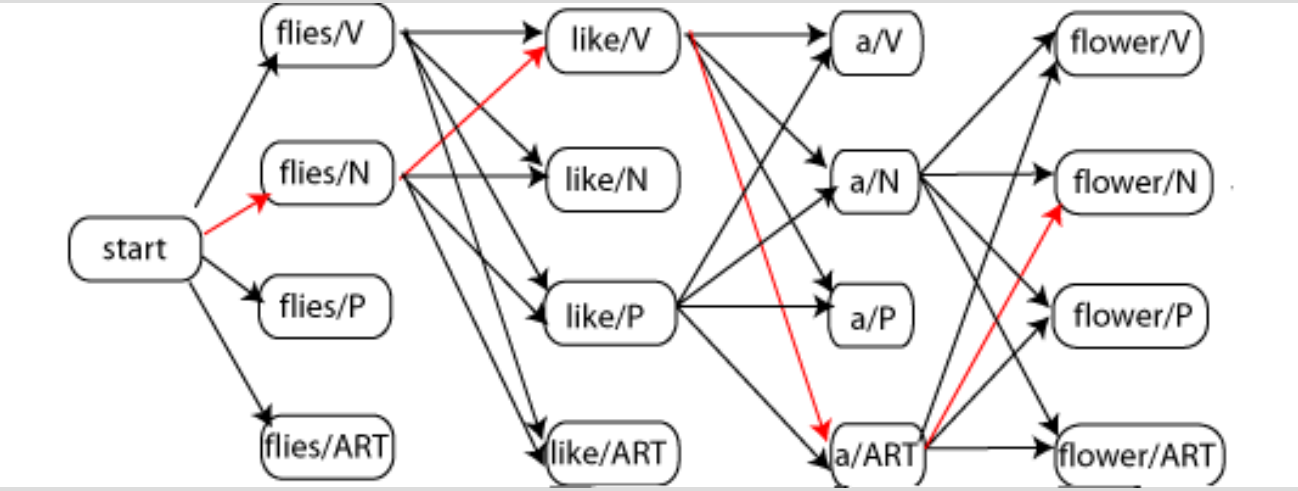
   (VP saw
   (NP John) (PP with (NP the telescope)))

Gramáticas probabilísticas en NLTK (7)

Ahora bien, el **algoritmo Viterbi es** un algoritmo de programación dinámica que permite hallar la secuencia más probable de estados ocultos (el llamado *camino de Viterbi*) que produce una secuencia observada de sucesos, especialmente en el contexto una cadena de Markov.

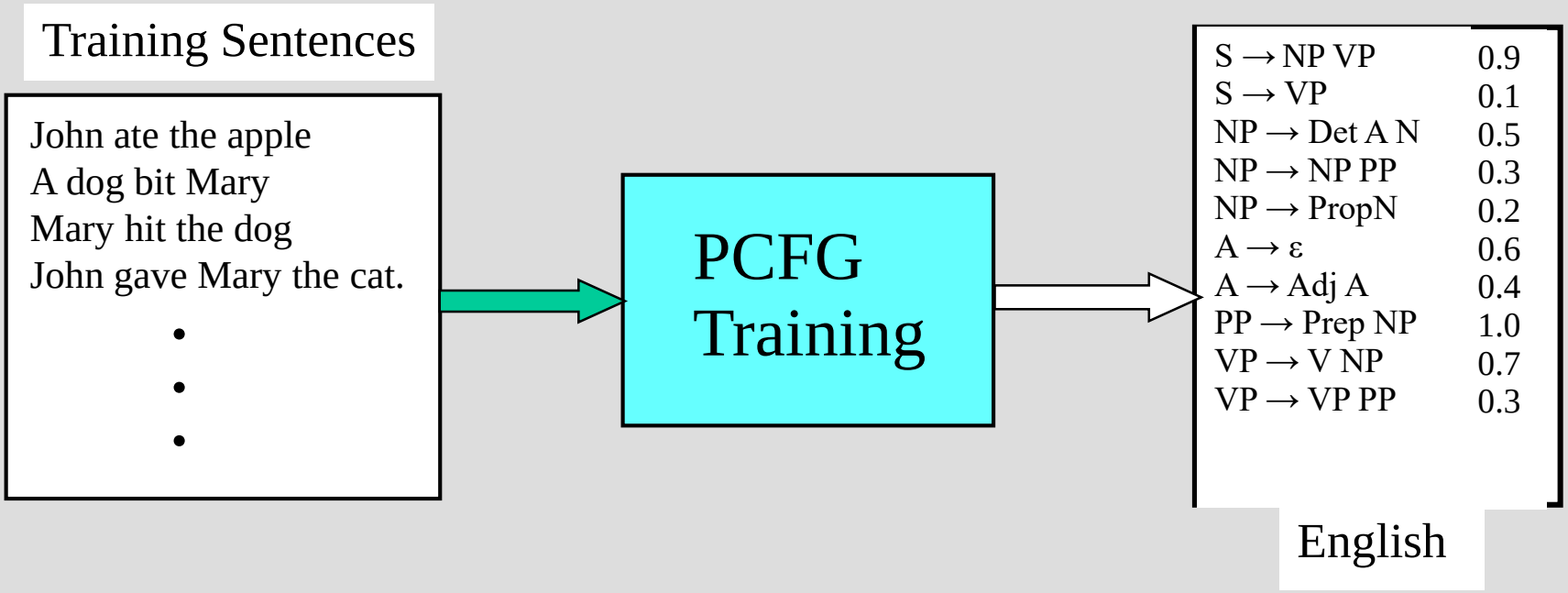


Andrew Viterbi
(1935)



Gramáticas probabilísticas en NLTK (8)

Aprovechando esta clase de recursos, podemos desarrollar programas y entrenarlos para que sean capaces de hacer análisis sintáctico en corpus, de modo que conforme vayan mejorando sus resultados, podríamos contar con un recurso sumamente útil para identificar patrones gramaticales regulares en una lengua natural:



Gracias por su atención

Blog del curso:
http://cesaraguilar.weebly.com/introduccioacuten-a-la-linguumliacutestica-computacional.html