PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

FACULTAD DE LETRAS

Introducción a la lingüística computacional

César Antonio Aguilar

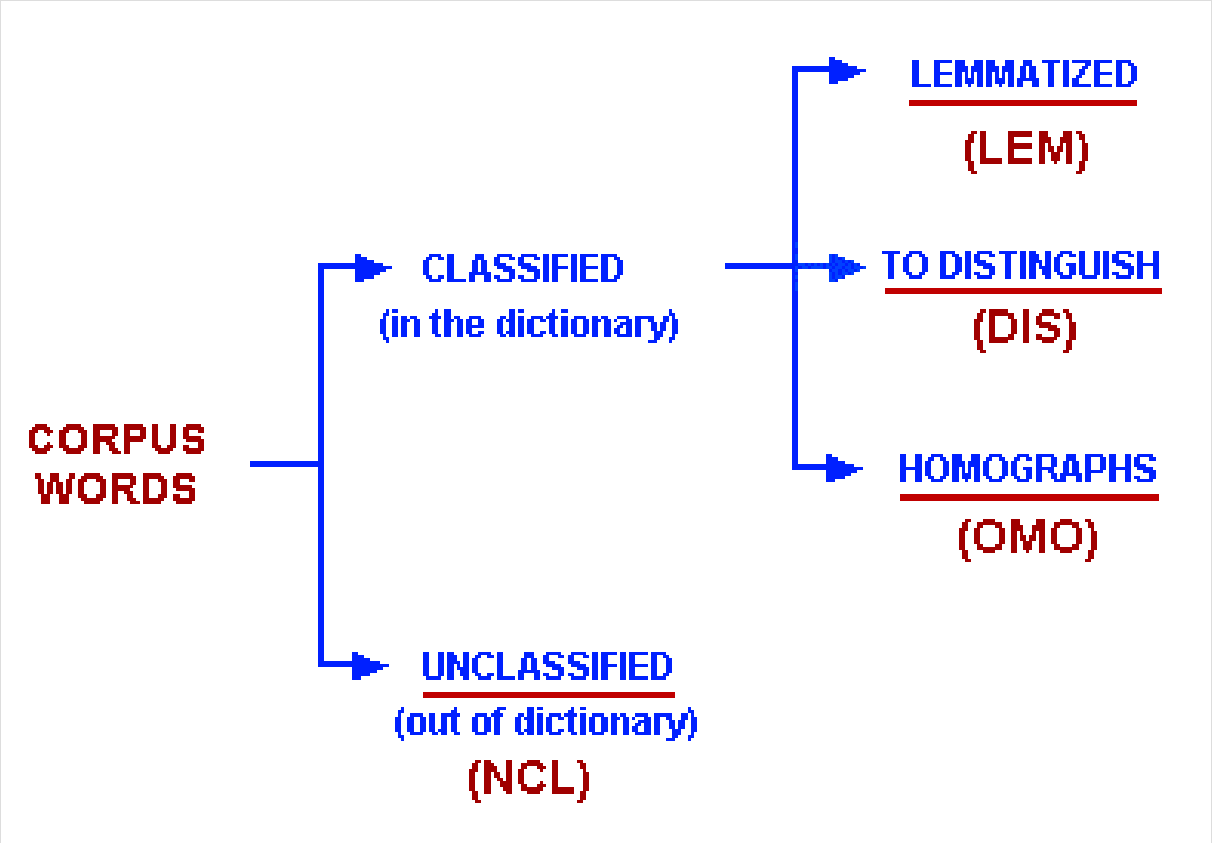
Facultad de Lenguas y Letras

03/10/2017

Cesar.Aguilar72@gmail.com

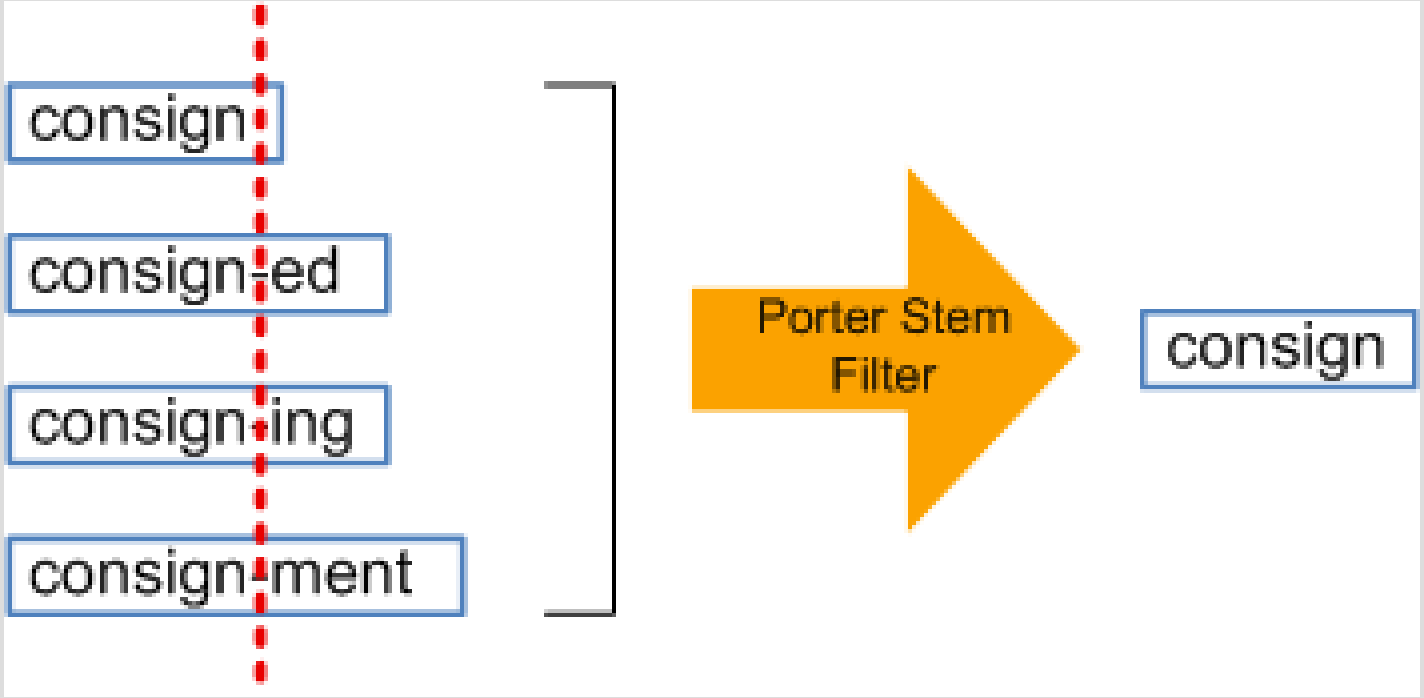
Síntesis de la clase pasada (1)

En la clase pasada vimos cómo puede realizarse un análisis de palabras aplicando un proceso denominado **lematización**, el cual consiste en asociar una forma de palabra aparecida en un texto con su forma canónica (o lematizada).



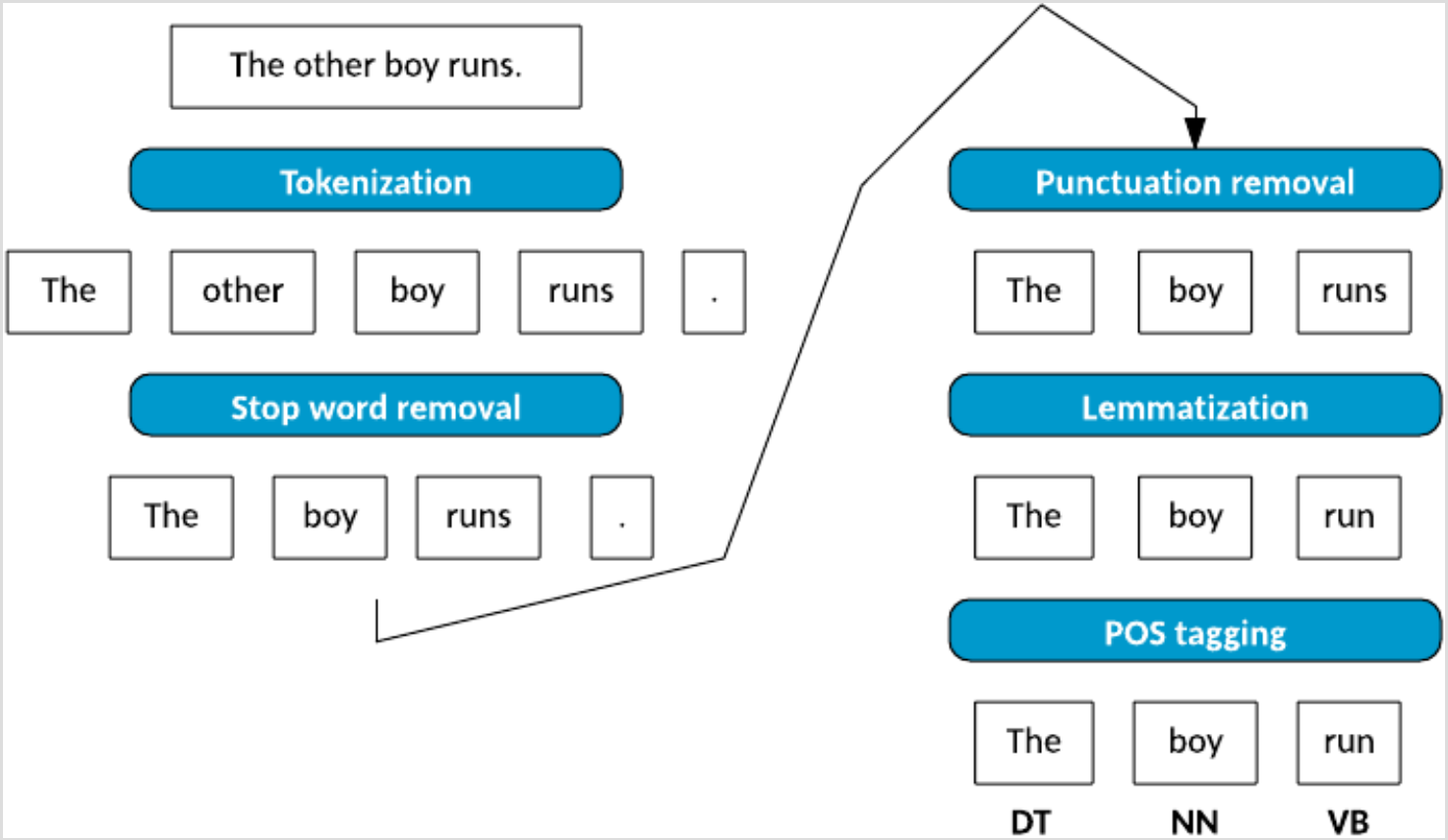
Síntesis de la clase pasada (2)

Para lograr esto, aplicamos el algoritmo de Porter, el cual nos permite identificar y segmentar los **sufijos** ligados a una **raíz**, de tal suerte que podemos separalos a partir de un corte de +/- 5 caracteres:



¿Por qué es importante la lematización? (1)

Cuando analizamos el vocabulario de un texto, además de tener una buena segmentación del mismo, requerimos que dicha segmentación nos ayude a reconocer y asociar de manera pertinente una etiqueta gramatical con su correspondiente forma de palabra.



¿Por qué es importante la lematización? (2)

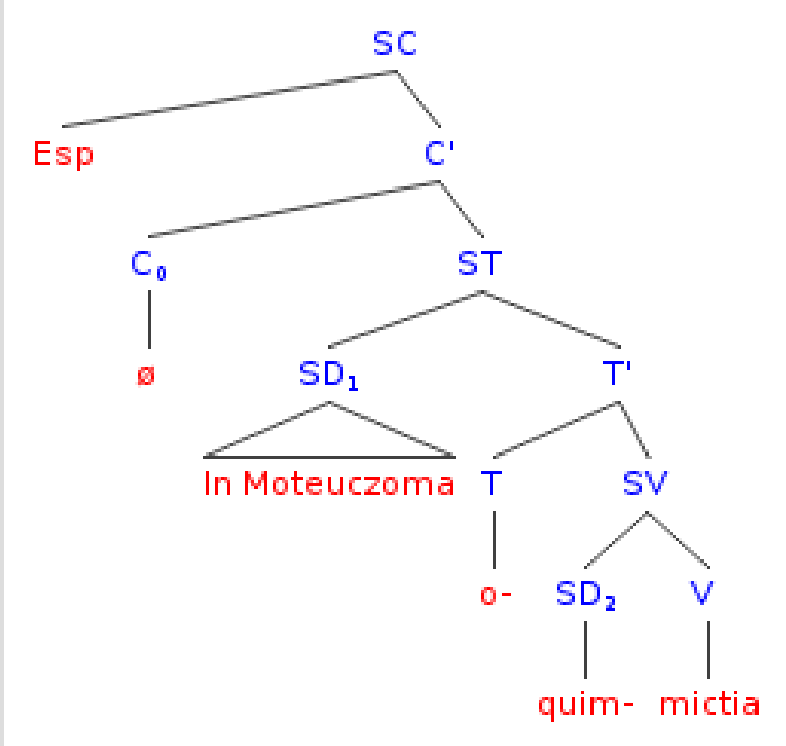
La lematización es una tarea que nos permite analizar computacionalmente el nivel morfológico de lenguas documentadas y con tradición escrita, tales como el inglés o el español.

Empero, no es la única vía para hacer esta clase de análisis. Un caso concreto: ¿cuál es la situación de las lenguas indígenas de América?

¿Por qué es importante la lematización? (3)

Algunas lenguas cuentan con registros históricos tales como el **náhuatl**, gracias al trabajo de misioneros como **Andrés de Olmos** u **Horacio Carochi**, autor de la primera gramática de esta lengua. Sin embargo, no siempre ocurre así con todas las lenguas.

ARTE
DE
LA LENGVA
MEXICANA
CON LA DECLARACION DE LOS
ADVERBIOS DELLA.
Al
Illustriss.º y Reuerendiss.º
Señor Don Iuan de Mañozca Arçobispo de
Mexico, del Consejo de su Magestad, &c.
POR EL PADRE HORACIO CAROCHI
Rector del Colegio de la Compañia de IESVS de San
Pedro, y San Pablo de Mexico.
Año de 1645.
IMPRESSO CON LICENCIA
En MEXICO: Por Iuan Ruyz. Año de 1645.



Morfología y entropía (1)

El uso de la entropía en lingüística computacional es sumamente útil para líneas de investigación como la morfología. Veamos un ejemplo, revisando el trabajo de un colega mexicano, Alfonso Medina (**El Colegio de México**), quien ha desarrollado un sistema capaz de detectar afijos en distintas lenguas, implementando un método no-supervisado.

http://cell.comex.mx/index.php/Alfonso-medina

Morfología y entropía (2)

Chuj (lengua perteneciente al grupo mayense)

Rarámuri (lengua perteneciente al grupo yutoazteca)

La idea de Alfonso es la siguiente: si bien algoritmos como el de Porter son excelentes para detectar las raíces y los sufijos de una lengua, muchas veces es insuficiente para determinar estas unidades en lenguas poco o nada documentadas. En México, existen muchos de estos casos:

Morfología y entropía (3)

Los sufijos, como toda unidad lingüística, están sometidos a dos relaciones concretas: las sintagmáticas, y las paradigmáticas:

Paradigm ↑

The	cat	sat	on	the	mat
A	dog	ran	in	the	park
The	man	got	on	a	bus

Syntagm →

En términos probabilísticos, Alfonso las describe como relaciones entre *acompañantes* y *alternantes*:

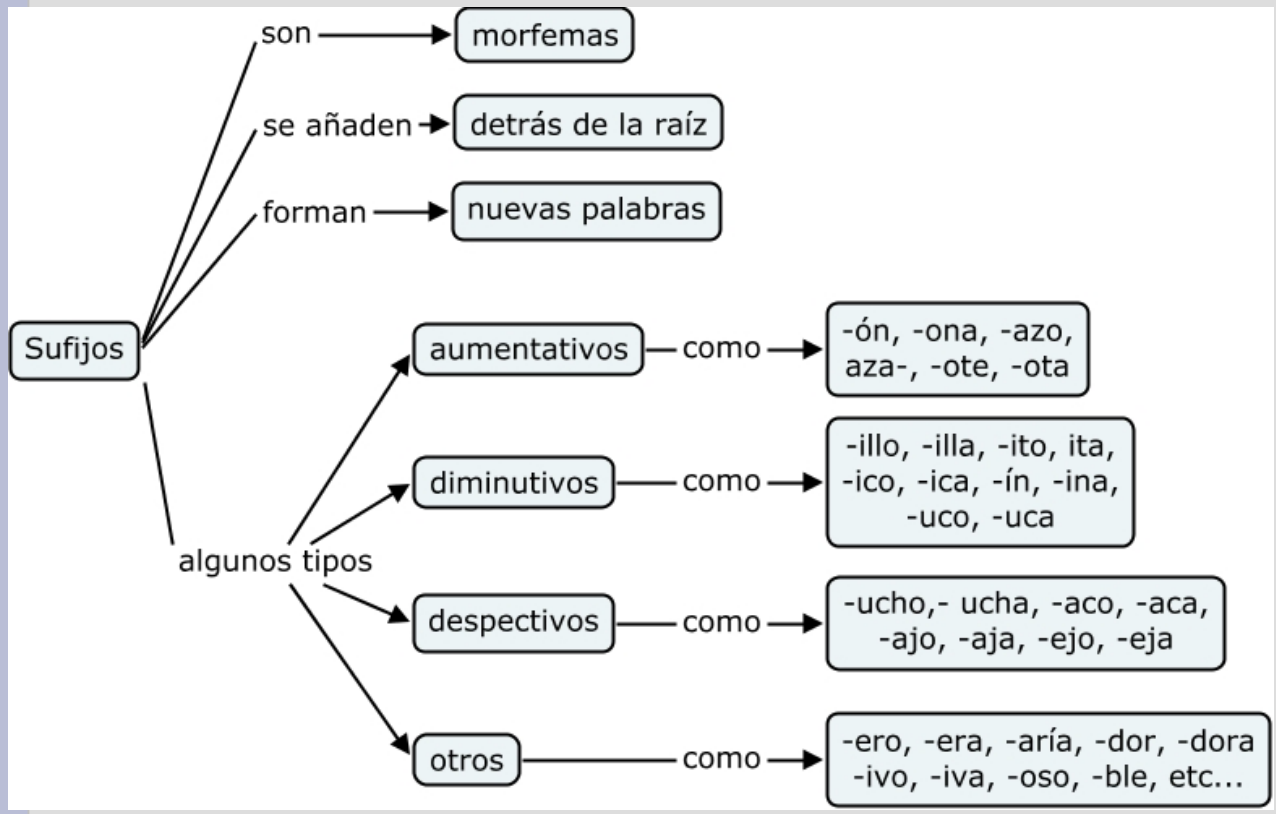
1. *companions* — word fragments which appear attached to the affix candidate (syntagmatic relation).

2. *alternants* — word fragments which occur in complementary distribution with the affix candidate.

Morfología y entropía (4)

$$k = \frac{companions}{alternants}$$

Esto se traduce en una fórmula como la que sigue, en donde *K* es un valor que describe qué tan fijo o cambiante puede ser una unidad morfológica dentro de una palabra.



Morfología y entropía (5)

La hipótesis de Alfonso es la siguiente: dadas una lista de raíces y sufijos de una lengua *X*, es posible identificar a los segundos *considerando su alto nivel de entropía, resultado de poseer un potencial combinatorio alto.*

TABLE 1
Catalog of Ralámuli Suffixes (top 30 candidates)

RANK	SUFFIX	FREC.	ECONOMY	ENTROPY	AFFIXALITY
1.	~ma	35	1.00000	0.88030	0.98050
2.	~re	77	0.81100	0.86060	0.82370
3.	~sa	33	0.93060	0.75590	0.77430
4.	~ra	62	0.64610	0.85080	0.71940
5.	~si	28	0.52570	0.83450	0.70340
6.	~na	25	0.72240	0.79840	0.64410
7.	~go	4	0.90650	0.64930	0.59000
8.	~é	49	0.43580	1.00000	0.53400
9.	~ame	51	0.30640	0.85910	0.47250
10.	~gá	18	0.37810	0.61360	0.46550
11.	~ka	19	0.28060	0.84130	0.45920
12.	~á	67	0.31330	0.91950	0.45710
13.	~ré	11	0.41020	0.73430	0.43780
14.	~ga	50	0.28340	0.80650	0.42430
15.	~a	281	0.18960	0.97250	0.42250
16.	~ba	8	0.30220	0.74000	0.41880
17.	~ayá	8	0.44320	0.57570	0.41110
18.	~í	42	0.26480	0.80540	0.39070
19.	~či	39	0.27510	0.74000	0.37260
20.	~e	164	0.29100	0.64290	0.36210
21.	~mi	4	0.30220	0.69910	0.35760
22.	~áame	12	0.00000	0.90570	0.35350
23.	~yá	20	0.11080	0.57420	0.34260
24.	~i	139	0.10320	0.84220	0.32820
25.	~ira	11	0.10990	0.79570	0.32350
26.	~o	41	0.00000	0.96810	0.32270
27.	~ne	3	0.40290	0.41950	0.32170
28.	~wa	9	0.13430	0.74000	0.30730
29.	~agá	6	0.20140	0.53640	0.30150
30.	~sí	4	0.00000	0.53740	0.29820

Morfología y entropía (6)

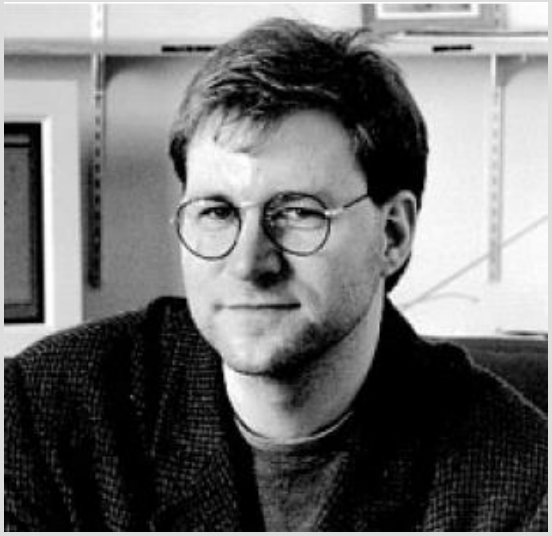
En contraste, las raíces se pueden reconocer *tomando en cuenta que tienen un nivel de entropía bajo, y por lo mismo tienen un poder combinatorio reducido.*

TABLE 6
Evaluation measurements

	RALÁMULI	CHUJ	
subsystem	derivational suffixes	prefix verbal inflection	suffix verbal inflection
sample tokens	3,584	15,485	
sample types	934	2,300	
subsystem size[a] (n)	35	20	12
right guesses[b]	28	15	12
presumed errors[b]	7	5	0
unretrieved items[c]	10	0	3
precision	0.80	0.75	1.00
recall	0.71	1.00	0.75

El algoritmo de Brill (1)

Otro algoritmo útil en esta clase de procesamiento al nivel de palabra es el que desarrolló Eric Brill, el cual reconoce formas de palabras, asociándolas posteriormente a una etiqueta sintáctica. Al respecto pueden ver el siguiente sitio WEB:



Eric Brill

http://en.wikipedia.org/wiki/Brill_tagger

El algoritmo de Brill (2)

El algoritmo de Brill se fundamenta en el aprendizaje basado en transformaciones y dirigido por el error. Este algoritmo consta de los siguientes módulos:

- Etiquetador léxico
- Etiquetador de palabras desconocidas
- Etiquetador contextual

El algoritmo de Brill (3)

El etiquetador léxico funciona del siguiente modo:

Cada palabra se etiqueta inicialmente con su etiqueta más probable

No se tiene en cuenta el contexto en el que aparece

Palabras desconocidas:

- sustantivo propio si la primera letra es mayúscula
- sustantivo común en otro caso

El algoritmo de Brill (4)

Por su parte, el etiquetador de palabras desconocidas intenta adivinar una etiqueta para una palabra desconocida en función de sus afijos (sufijos o prefijos), y de otras propiedades relevantes similares

Cada transformación consta de dos partes

- una descripción del contexto de aplicación
- una regla de reescritura que reemplaza una etiqueta por otra

El algoritmo de Brill (5)

El algoritmo de Brill cuenta también con una plantilla genérica de transformaciones léxicas, esto es:

- `A` y `B` son variables sobre el conjunto de todas las etiquetas
- `x` es cualquier cadena de caracteres de longitud 1, 2, 3 o 4
- `l` es la longitud de dicha cadena
- `w` es cualquier palabra
- `z` es cualquier caracter.

El algoritmo de Brill (6)

El etiquetador contextual actúa justo después del etiquetador de palabras desconocidas, ya que aplica en orden una secuencia de reglas contextuales que han sido aprendidas de manera automática a partir del corpus de entrenamiento. Un ejemplo es:

- A, B, C y D son variables sobre el conjunto de todas las etiquetas
- w y x son cualquier palabra

El algoritmo de Brill (7)

Veamos un caso aplicado al inglés. Supongamos que queremos aplicar un proceso de etiquetado basados en el Corpus Brown:

- Etiquetas (corpus Brown):
  - VBN: verbo en participio pasado
  - VBD: verbo en tiempo pasado
  - NP: sustantivo propio
  - BEDZ: palabra was
  - BY: palabra by
  - PPS: pronombre nominativo singular en tercera persona
- VBN y VBD son las etiquetas más probables para las palabras killed y shot, respectivamente

El algoritmo de Brill (8)

Aplicamos nuestro etiquetador, y obtenemos oraciones anotadas como:

(1) Chapman/NP killed/VBN John/NP Lennon/NP
(2) John/NP Lennon/NP was/BEDZ shot/VBD by/BY Chapman/NP

(3) He/PPS witnessed/VBD Lennon/NP killed/VBN by/BY Chapman/NP

- Errores:

en (1) killed etiquetada incorrectamente como verbo en participio pasado

en (2) shot incorrectamente etiquetada como verbo en tiempo pasado

El algoritmo de Brill (9)

Si aplicamos el algoritmo de Brill, el módulo de etiquetado contextual podría aplicar las siguientes reglas:

VBN VBD prevtag NP (*cambiar la etiqueta* VBN *por* VBD *si la etiqueta previa es* NP)

VBD VBN nexttag BY (*cambiar* VBD *por* VBN *si la siguiente etiqueta es* BY)

- Al aplicar la primera regla la palabra killed que aparece en las frases (1) y (3) cambia su etiqueta VBN por VBD:

(4) Chapman/NP killed/VBD John/NP Lennon/NP
(5) John/NP Lennon/NP was/BEDZ shot/VBD by/BY Chapman/NP
(6) He/PPS witnessed/VBD Lennon/NP killed/VBD by/BY Chapman/NP

El algoritmo de Brill (10)

Aplicando la segunda regla obtenemos:

la palabra shot de la frase (5) cambia su etiqueta VBD por VBN
la palabra killed de la frase (6) vuelve a cambiar su etiqueta VBD otra vez por VBN

(7) Chapman/NP killed/VBD John/NP Lennon/NP
(8) John/NP Lennon/NP was/BEDZ shot/VBN by/BY Chapman/NP
(9) He/PPS witnessed/VBD Lennon/NP killed/VBN by/BY Chapman/NP

Para mayores detalles, pueden consultar la siguiente página, la cual es una explicación que dan Miguel Alonso, Jorge Graña y Jesús Virales, de la Universidad de La Coruña (España):

http://www.grupolys.org/docencia/ln/biblioteca/traspas_brill.pdf

Gracias por su atención

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