Introducción al Procesamiento de Lenguaje Natural

Grupo PLN – InCo

Lenguajes

¿Qué es un lenguaje?

(1) Capacidad propia del ser humano para expresar pensamientos y sentimientos por medio de palabras

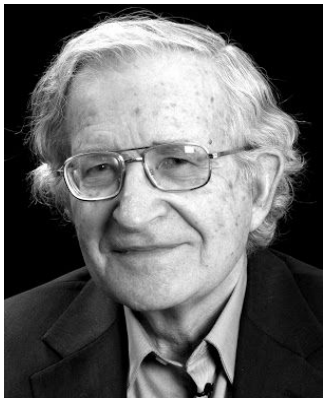
(2) Sistema de signos que utiliza una comunidad para comunicarse oralmente o por escrito.

(3) Sistema de comunicación estructurado para el que existe un contexto de uso y ciertos principios combinatorios formales

Lenguajes

¿Qué es un lenguaje?

Conjunto finito o infinito de oraciones, cada una de las cuales posee una extensión finita, construida a partir de un conjunto finito de elementos (Chomsky 1957)



Lenguajes

- Alfabeto
- Reglas

```
' Globales -----------
Var Variable0:Booleano
Var Variable1:Cadena
' Fin Globales -------
Proc Procedimiento ' <- Procedimiento sin retorno.
   Var Variable2:Entero ' Locales
   Var Variable3:Real

   Si Variable0 = Falso Entonces ' Condición "If"
      Contar Variable2 = 0 a 9 ' Bucle "For"
         Variable1 = Variable1 + "1"
      Seguir ' "End For"
   FinSi ' "End If"

   Variable3 = 5.13
FinProc
```

```
>>> for i, v in enumerate(['tic', 'tac', 'toe']):
...     print i, v
...
0 tic
1 tac
2 toe
```

0 → SN SV

SN → DET N

SV → V COMP

La casa es preciosa

```
# <> isn't actually a valid comparison operator in Python. It's here for the
# sake of a __future__ import described in PEP 401 (which really works :-)
comp_op: '<'|'>'|'=='|'>='|'<='|'<>'|'!='|'in'|'not' 'in'|'is'|'is' 'not'
star_expr: '*' expr
expr: xor_expr ('|' xor_expr)*
xor_expr: and_expr ('^' and_expr)*
and_expr: shift_expr ('&' shift_expr)*
shift_expr: arith_expr (('<<'|'>>') arith_expr)*
arith_expr: term (('+'|'-') term)*
term: factor (('*'|'@'|'/'|'%'|'//') factor)*
factor: ('+'|'-'|'~') factor | power
```

Lenguajes

- Formales
  - Definidos por reglas pre-establecidas
- Naturales
  - Evolucionan con el tiempo
  - Utilizados para la comunicación humana
  - Las reglas “se desarrollan” después que sucede el hecho

Lenguajes Regulares

English is not a finite state language.
(Chomsky 1957)

Motivación

- Lenguaje para identificar strings de caracteres (Kleene, 1956)
- Herramienta para especificar e identificar textos mediante patrones
- Expresividad limitada, pero *muy* eficientes.

Motivación

Se requiere de:

- Patrón (***qué*** se quiere buscar)
- Corpus (***dónde*** se quiere buscar)

Luego, una función de búsqueda recorre el corpus devolviendo todas las ocurrencias del patrón

Historia

- Los lenguajes regulares están completamente estudiados, podrían considerarse un problema resuelto
- Desde Unix (1969) están incorporados a los lenguajes de programación
- Todos los lenguajes de programación hoy los incluyen como la forma de especificar patrones

Expresiones Regulares

Notación formal para definir lenguajes regulares sobre un alfabeto Σ

- $\varnothing$ es una ER que describe al conjunto $\varnothing$

- **a** es una ER $\forall a \in \Sigma \cup \{\varepsilon\}$

- Si r y s son ER para describir R y S respectivamente entonces:
  - (r|s) es una ER para $R \cup S$, *unión*
  - (r.s) es una ER para R.S, *concatenación*
  - (r*) es una ER para R*, *clausura de Kleene*

Estos son todas las Expresiones Regulares definidas sobre Σ

ER y Búsquedas

Algunos ejemplos:

ER	Patrón encontrado
cabeza	de la **cabeza** al sombrero
!	Vení para acá**!**
[vV]agón	en un **vagón** cargado de sandías
[abc]	solo de no**c**he
[0-9]	Capítulo **1**: Introducción
[^A-Z]	M**a**ñana va a ser un gran día
[^sS]	**T**engo permiso

¿Cómo hacemos para buscar cualquiera de las ocurrencias de oveja?

oveja

Oveja

ER y Búsquedas

Algunos ejemplos:

ER	Patrón encontrado
cabeza	de la **cabeza** al sombrero
!	Vení para acá**!**
[vV]agón	en un **vagón** cargado de sandías
[abc]	solo de no**c**he
[0-9]	Capítulo **1**: Introducción
[^A-Z]	M**a**ñana va a ser un gran día
[^sS]	**T**engo permiso

¿Cómo hacemos para buscar cualquiera de las ocurrencias de oveja?

oveja

Oveja

[oO]veja

ER y Búsquedas

Más ejemplos:

ER	Patrón encontrado
[e^]	símbolo **^** significa que....
a^b	cuando encuentres "**a^b**" es
casas?	Juan había ido a **casa** de
colour?r	un **color** one **colour**
pec.s	tenía **pecas** en toda la cara
/beee*!/	La oveja hace **beeeeee!**
/bee+!/	La oveja hace **beeeeee!**
/^.*Juan vino\..*$/	**Carlos cerveza. Juan vino. José agua.**
/\bel\b/	Caminaba por **el** camino de
	(*) habían elementos arriba

ER y Búsquedas

Precedencia de operadores:

```
    ( )
* + ? { }
```

Secuencia de caracteres y “anclado”

Disyunción |

Ejemplo:

/\b[0-9]+\s*(Mhz|[Mm]egahertz | Ghz|[Gg]igahertz)\b/

ER y Búsquedas

Operadores avanzados

\d → cualquier dígito [0-9]
\w → cualquier carácter alfanumérico o "espacio"
\s → [\b\t\n]
y las mayúsculas son la negación

{n} → **n** ocurrencias del carácter o expresión anterior
{n,m} → de **n** a **m** ocurrencias " "
{n,} → al menos **n** ocurrencias " "

ER y Sustituciones

Sustitución: sustituye un patrón por otro

```
s / ( [0-9]+ ) / <\1> /
```

donde el \1 como segundo patrón, refiere al primero

... tenía el número 10 en la espalda ...
x
... tenía el número <10> en la espalda ...

ER y Sustituciones

Paréntesis y operadores, pueden ser usados para especificar ciertas expresiones a sustituir

Ejemplo:

(1) hoy (.*)ro estar aquí, mañana \1ro morir

– hoy quiero estar aquí, mañana quiero morir

NO – hoy quiero estar aquí, mañana prefiero morir

(2) hoy (.*)ro estar (.*), mañana \1ro morir \2

– hoy quiero estar aquí, mañana quiero morir aquí

NO – hoy quiero estar aquí, mañana quiero morir allí

Este tipo de facilidades son extensiones de las ER

ER y Sustituciones

ELIZA: programa que simula una conversación entre un paciente y un psicólogo (Weizembaum, 1966)

Paciente: Men are all alike
ELIZA: IN WHAT WAY
Paciente: They're always bugging us about something or other
ELIZA: CAN YOU THINK OF A SPECIFIC EXAMPLE
Paciente: Well, my boyfriend made me come here
ELIZA: YOUR BOYFRIEND MADE YOU COME HERE
Paciente: He says I'm depressed much of the time
ELIZA: I AM SORRY TO HEAR YOU ARE DEPRESSED

ER y Sustituciones

Algunas sustituciones usadas:

s/ .* I AM (depressed|sad) .*/ I AM SORRY TO HEAR YOU ARE \1 /

s/ .* MY (mother | father | boyfriend | girlfriend | friend) MADE ME COME HERE .*/ WHY DO YOU THINK YOUR \1 YOU COME HERE /

s/ .* all .*/ IN WHAT WAY /

s/ .* always .*/ CAN YOU THINK OF A SPECIFIC EXAMPLE /

Lenguajes Regulares

Conjunto de strings denotados por *expresiones regulares* definidas sobre un alfabeto Σ

$L = L(r)$

Operaciones de clausura:

intersección $(L_1 \cap L_2)$
diferencia $(L_1 - L_2)$
complemento $(\Sigma^* - L_1)$
reverso (L_1^r)

Autómatas Finitos

Máquina de estados

- En particular, nos interesan desde el punto de vista del reconocimiento de ***Lenguajes Regulares***

- Un *lenguaje regular* es el conjunto de strings sobre un alfabeto Σ reconocidos por *autómatas finitos*

Autómatas Finitos

Autómata Finito Determinista $M:(Q, \Sigma, \Delta, q_0, F)$

Q: conjunto finito de *n* estados
Σ: alfabeto finito de símbolos de entrada
q_0: estado inicial
F: conjunto de estados finales (o aceptores) $F \subseteq Q$
$\Delta: Q \times \Sigma \rightarrow Q$
$\Delta(q_i, a) = q_j$ función de transición entre estados

Autómatas Finitos

Ejemplo: el lenguaje de las ovejas

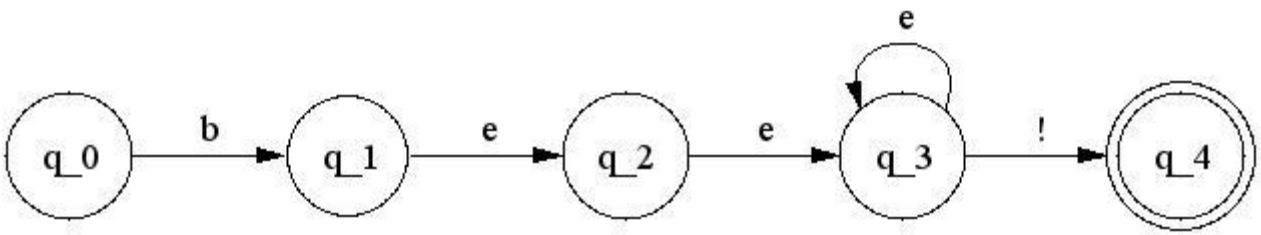
lo podemos ver como secuencias (infinitas) de tiras del tipo

bee!
beee!
beeee!
beeeee!
...

donde la ER asociada sería /bee+!/

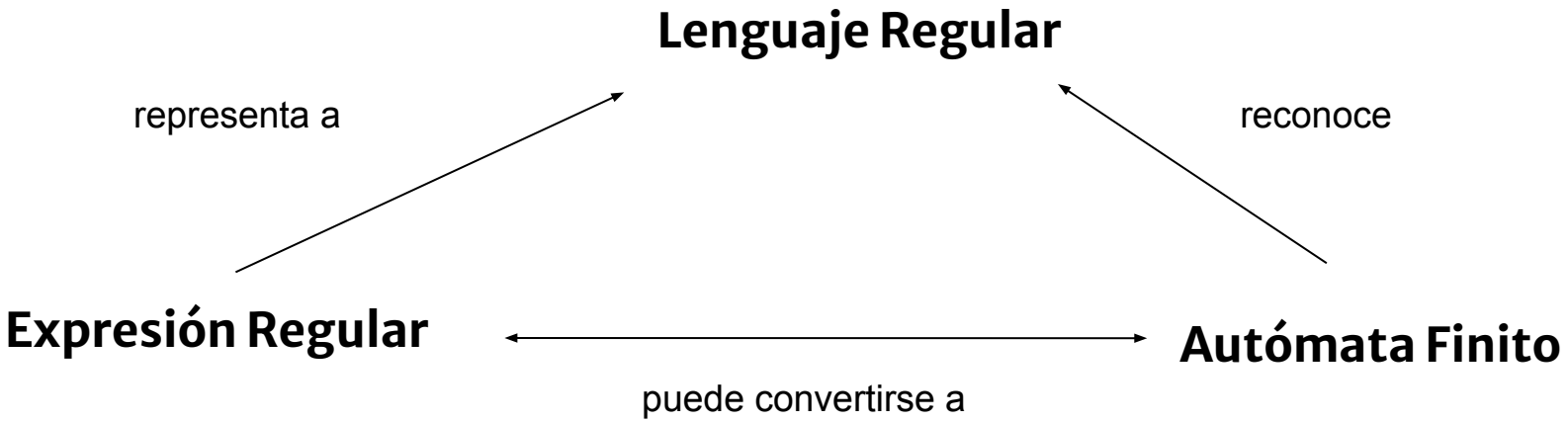
Autómatas Finitos

y el autómata finito....



Autómatas Finitos

Cualquier ER puede ser "implementada" por un AF y recíprocamente

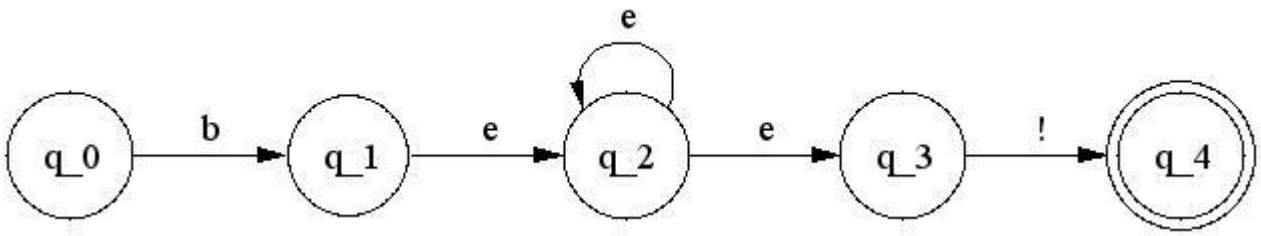


Autómatas Finitos

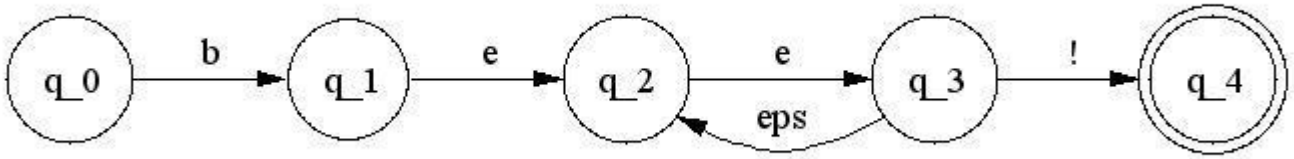
```
funcion D-Reconocimiento(buffer,automata) : SI/NO
   i := comienzo del buffer(tira)
   estado_actual := q_0
   repetir
      si fin_de_tira entonces
         si estado_actual ∈ F entonces
            retornar(SI)
         sino
        retornar(NO)
      sino
         si existe(delta(estado_actual,buffer[i])) entonces
                estado_actual := delta(estado_actual,buffer[i])
                 i := i + 1
         sino
         retornar(NO)
   fin repetir
```

Autómatas Finitos

Autómata Finito NO Determinista



Autómata Finito NO Determinista-ε



Autómatas Finitos

- Los AFD y los AFND reconocen lenguajes equivalentes. (Hopcroft 1979)
- A veces la construcción de un AFND para reconocer un lenguaje determinado es más fácil que construir un AFD
- El AFND se puede utilizar para reducir la complejidad del trabajo matemático para demostrar muchas propiedades importantes en la teoría de la computación
- PERO, el costo es el espacio…
- En los AFND, el problema es "elegir" el camino adecuado para procesar la tira

Lenguajes formales vs naturales

- Formales
  - Definidos por reglas pre-establecidas
- Naturales
  - Evolucionan con el tiempo
  - Utilizados para la comunicación humana
  - Las reglas "se desarrollan" después que sucede el hecho

Lenguajes formales vs naturales

- Los lenguajes naturales son los que “la gente habla”
- Es muy difícil modelar con un lenguaje formal al lenguaje natural, porque:
    - son ambiguos
    - son vagos
    - dependen del contexto

sin embargo... se puede discutir (J.M.Dunn, 1969)

ER y Lenguaje Natural

- El problema del *center embedding*

Un hombre llora
Un hombre que una mujer ama llora
Un hombre que una mujer que un niño duerme ama llora
…

$GN_1\ V_1$
$GN_1\ GN_2\ V_2\ V_1$
$GN_1\ GN_2\ GN_3\ V_3\ V_2\ V_1$

ER y Lenguaje Natural

- No podemos modelar el lenguaje natural con expresiones regulares (¿no?)

- Podemos modelar algunas cosas:
  - Fonología
  - Morfología
  - Sintaxis (algo)

Referencias

- J.Martin & D.Jurafsky. Speech and Language Processing – Capítulo 2

- Hopcroft & Ullman: Int. To Automata Theory, Languages and Computation