PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

FACULTAD DE LETRAS

Introducción a la lingüística computacional

César Antonio Aguilar

Facultad de Lenguas y Letras

31/08/2017

Cesar.Aguilar72@gmail.com

Ejercicio de la clase pasada (1)

Antes de empezar la clase de hoy, vamos a terminar el ejercicio de la clase pasada que teníamos pendiente, que era analizar un nuevo texto con las herramientas de la sub-libería **nltk.book**. Para eso, vamos a ir al siguiente sitio WEB:

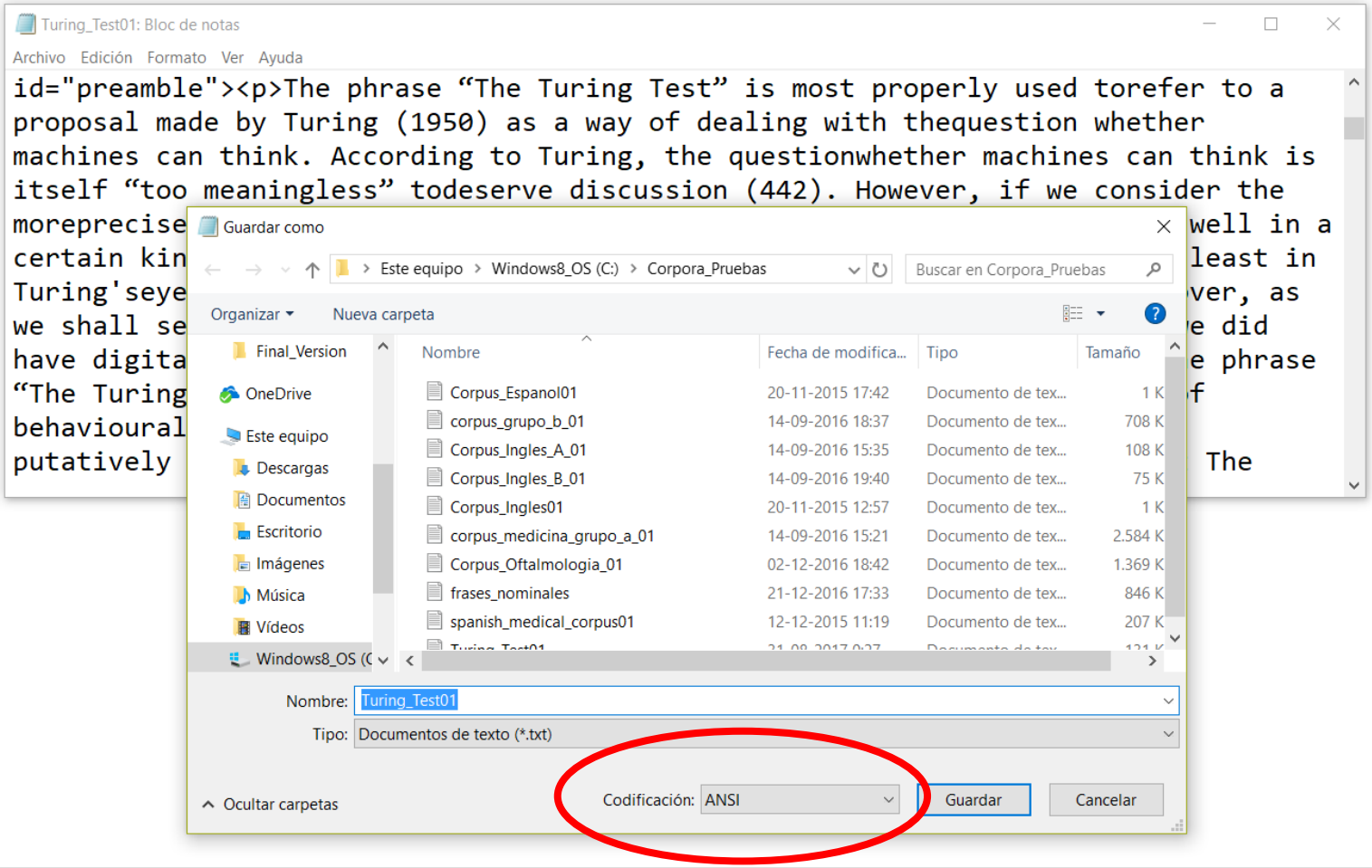
https://plato.stanford.edu/entries/turing-test/

Esto es una entrada de la **Stanford Encyclopedia of Philosophy**:

Stanford Encyclopedia of Philosophy

Ejercicio de la clase pasada (2)

El problema que tuvimos en la sesión pasada era reconocer correctamente la codificación de nuestro texto. Una forma de resolver este problema sería guardar esta página como un documento con extensión TXT, y utilizar una codificación en ANSI, como se ve en la imagen:



Ejercicio de la clase pasada (3)

El código que necesitamos para procesar nuestro documento codificado en ASCI es el siguiente:

```
import nltk, re

Cadena_Turing01 = open('C://Corpora_Pruebas/Turing_Test01.txt',
'rU').read()

Tokens_Turing01 = nltk.word_tokenize(Cadena_Turing01)
```

Con esto obtenemos:

```
In [9]: type(Tokens_Turing01)
Out[9]: list

In [10]: len(Tokens_Turing01)
Out[10]: 27904
```

Ejercicio de la clase pasada (4)

Empero, nuestra lista de tokens necesita una buena limpieza, ya que hay muchos elementos que son basura, veamos:

```
In [7]: Rango_Turing01 = Tokens_Turing01[0:200]

In [8]: print(Rango_Turing01)
['<', '!', 'DOCTYPE', 'html', '>', '<', '!', '--', 'saved', 'from', 'url=', '(', '0047', ')', 'https', ':',
'//plato.stanford.edu/entries/turing-test/', '--', '>', '<', 'html', '>', '<', '!', '--', '<', '!', '[',
'endif', ']', '--', '>', '<', 'head', '>', '<', 'meta', 'http-equiv=', "'", 'Content-Type', "'",
'content=', "'", 'text/html', ';', 'charset=UTF-8', "'", '>', '<', 'meta', 'name=', "'", 'viewport',
"'", 'content=', "'", 'width=device-width', ',', 'initial-scale=1.0', "'", '>', '<', 'title', '>',
'The', 'Turing', 'Test', '(', 'Stanford', 'Encyclopedia', 'of', 'Philosophy', ')', '<', '/title', '>', '<',
'meta', 'name=', "'", 'robots', "'", 'content=', "'", 'noarchive', ',', 'noodp', "'", '>', '<', 'meta',
'property=', "'", 'citation_title', "'", 'content=', "'", 'The', 'Turing', 'Test', "'", '>', '<',
'meta', 'property=', "'", 'citation_author', "'", 'content=', "'", 'Oppy', ',', 'Graham', "'", '>',
'<', 'meta', 'property=', "'", 'citation_author', "'", 'content=', "'", 'Dowe', ',', 'David', "'", '>',
'<', 'meta', 'property=', "'", 'citation_publication_date', "'", 'content=', "'", '2003/04/09', "'",
'>', '<', 'meta', 'name=', "'", 'DC.title', "'", 'content=', "'", 'The', 'Turing', 'Test', "'", '>',
'<', 'meta', 'name=', "'", 'DC.creator', "'", 'content=', "'", 'Oppy', ',', 'Graham', "'", '>', '<',
'meta', 'name=', "'", 'DC.creator', "'", 'content=', "'", 'Dowe', ',', 'David', "'", '>', '<', 'meta',
'name=', "'", 'DCTERMS.issued', "'", 'content=', "'", '2003-04-09', "'", '>', '<', 'meta', 'name=',
"'", 'DCTERMS.modified', "'", 'content=', "'", '2016-02-08', "'", '>']
```

Ejercicio de la clase pasada (5)

De hecho, es hasta que fijamos nuestro rango en la palabra 1900 de nuestra lista de tokens que podemos ver algo legible:

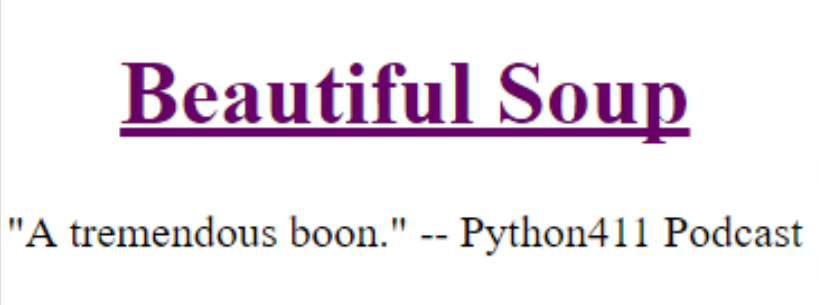
```
In [23]: Rango_Turing02 = Tokens_Turing01[1900:2100]

In [24]: print(Rango_Turing02)
['kind', 'of', 'game', 'that', 'Turing', 'describes', '(', '“The', 'Imitation', 'Game”', ')', ',', 'then–
at', 'least', 'in', "Turing's", 'eyes–we', 'do', 'have', 'a', 'question', 'that', 'admits', 'of',
'precise', 'discussion', '.', 'Moreover', ',', 'as', 'we', 'shall', 'see', ',', 'Turing', 'himself',
'thought', 'that', 'it', 'would', 'not', 'be', 'too', 'long', 'before', 'we', 'did', 'have', 'digital',
'computers', 'that', 'could', '“do', 'well”', 'in', 'the', 'Imitation', 'Game.', '<', '/p', '>', '<', 'p',
'>', 'The', 'phrase', '“The', 'Turing', 'Test”', 'is', 'sometimes', 'used', 'more', 'generally', 'to',
'refer', 'to', 'some', 'kinds', 'of', 'behavioural', 'tests', 'for', 'the', 'presence', 'of', 'mind', ',',
'or', 'thought', ',', 'or', 'intelligence', 'in', 'putatively', 'minded', 'entities', '.', 'So', ',',
'for', 'example', ',', 'it', 'is', 'sometimes', 'suggested', 'that', 'The', 'Turing', 'Test', 'is',
'prefigured', 'in', 'Descartes', "'", '<', 'em', '>', 'Discourse', 'on', 'the', 'Method', '<', '/em', '>',
'.', '(', 'Copeland', '(', '2000:527', ')', 'finds', 'an', 'anticipation', 'of', 'the', 'test', 'in',
'the', '1668', 'writings', 'of', 'the', 'Cartesian', 'de', 'Cordemoy', '.', 'Gunderson', '(', '1964', ')',
'provides', 'an', 'early', 'instance', 'of', 'those', 'who', 'find', 'that', 'Turing', "'s", 'work', 'is',
'foreshadowed', 'in', 'the', 'work', 'of', 'Descartes', '.', ')', 'In', 'the', '<', 'em', '>', 'Discourse',
'<', '/em', '>', ',', 'Descartes', 'says', ':', '<', '/p', '>', '<', 'blockquote', '>', 'If', 'there',
'were', 'machines', 'which', 'bore', 'a', 'resemblance']
```

La librería *Beautiful Soup* (1)

Un método para resolver el problema de la limpieza de nuestros documentos con formato HTML es hace uso de la librería **Beautiful Soup**, la cual nos permite editar esta clase de documentos desde Python —incluida nuestra plataforma Anaconda—, en aras de ejecutar tareas de extracción de información de la WEB (en inglés: *web scraping*).

Para más detalles sobre **Beautiful Soup**, pueden consultar el siguiente sitio:

www.crummy.com/software/BeautifulSoup/



La librería *Beautiful Soup* (2)

Ocupar **Beautiful Soup** es sencillo. Primero, necesitamos importar la librería **urllib**, junto con la función **request:**

```
import urllib.request
```

El siguiente paso es crear una variable a la cual le asignamos la dirección de la página WEB que queremos analizar. Para ello, usamos las siguientes instrucciones:

```
url01 = "https://plato.stanford.edu/entries/turing-test/"

Pre_Turing01 = urllib.request.urlopen(url01).read()
```

La librería *Beautiful Soup* (3)

Ahora, vamos a emplear **Beautiful Soup**, que importamos con la siguiente instrucción:

```
from bs4 import BeautifulSoup
```

Transformemos entonces el documento que se encuentra en nuestra URL en una cadena de caracteres:

```
Cadena_Turing01 = BeautifulSoup(Pre_Turing01).get_text()
```

Y obtenemos:

```
type(Cadena_Turing01)
str

len(Cadena_Turing01)
112039
```

La librería *Beautiful Soup* (4)

Lo que resta es transformar nuestra cadena en una auténtica lista de tokens:

```
Tokens_Turing02 = nltk.word_tokenize(Cadena_Turing01)
```

Esta lista, si la comparamos con la anterior, es mucho más clara:

```
In [11]: Rango_Turing02 = Tokens_Turing02[0:100]

In [12]: print(Rango_Turing02)
['The', 'Turing', 'Test', '(', 'Stanford', 'Encyclopedia', 'of', 'Philosophy', ')', 'Stanford', 'Encyclopedia',
'of', 'Philosophy', 'Menu', 'Browse', 'Table', 'of', 'Contents', 'What', "'s", 'New', 'Random', 'Entry',
'Chronological', 'Archives', 'About', 'Editorial', 'Information', 'About', 'the', 'SEP', 'Editorial', 'Board',
'How', 'to', 'Cite', 'the', 'SEP', 'Special', 'Characters', 'Advanced', 'Tools', 'Contact', 'Support', 'SEP',
'Support', 'the', 'SEP', 'PDFs', 'for', 'SEP', 'Friends', 'Make', 'a', 'Donation', 'SEPIA', 'for', 'Libraries',
'Entry', 'Navigation', 'Entry', 'Contents', 'Bibliography', 'Academic', 'Tools', 'Friends', 'PDF', 'Preview',
'Author', 'and', 'Citation', 'Info', 'Back', 'to', 'Top', 'The', 'Turing', 'TestFirst', 'published', 'Wed', 'Apr',
'9', ',', '2003', ';', 'substantive', 'revision', 'Mon', 'Feb', '8', ',', '2016', 'The', 'phrase', '“The',
'Turing', 'Test”', 'is', 'most', 'properly']
```

La librería *Beautiful Soup* (5)

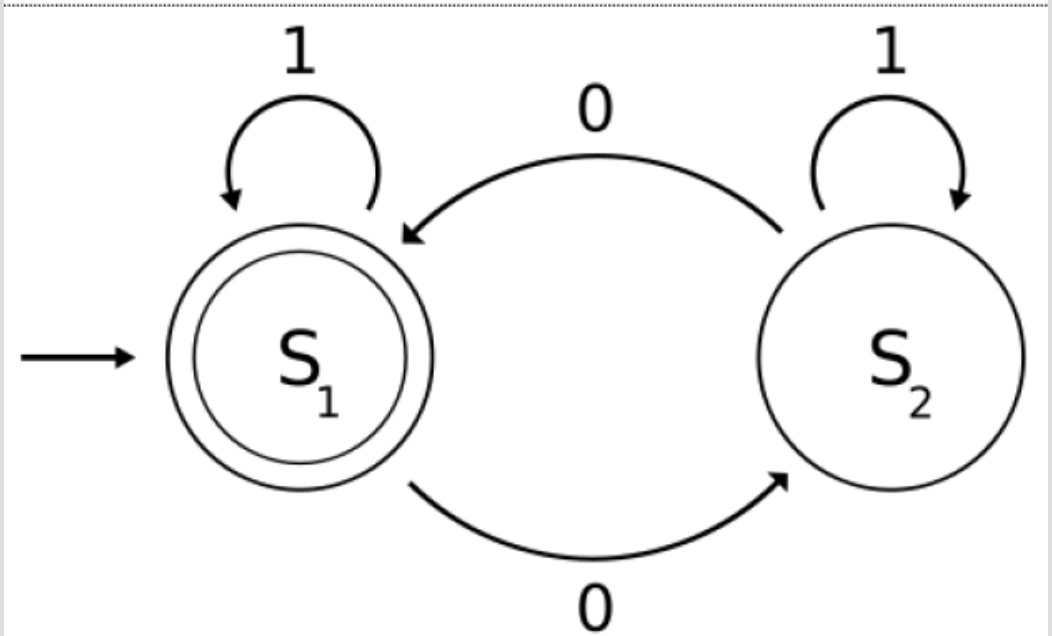
De hecho, si establecemos un rango a partir del token 1900, cambia por entero el resultado respecto a nuestro primer intento:

```
In [13]: Rango_Turing02 = Tokens_Turing02[1900:2100]

In [14]: print(Rango_Turing02)
['A', 'quick', 'look', 'at', 'the', 'transcripts', 'of', 'the', 'participants', 'for', 'the', 'preceding',
'decade', 'reveals', 'that', 'the', 'entered', 'programs', 'were', 'all', 'easily', 'detected', 'by', 'a',
'range', 'of', 'not-very-subtle', 'lines', 'of', 'questioning', '.', 'Moreover', ',', 'major', 'players', 'in',
'the', 'field', 'regularly', 'claimed', 'that', 'the', 'Loebner', 'Prize', 'Competition', 'was', 'an',
'embarrassment', 'precisely', 'because', 'we', 'were', 'still', 'so', 'far', 'from', 'having', 'a', 'computer',
'programme', 'that', 'could', 'carry', 'out', 'a', 'decent', 'conversation', 'for', 'a', 'period', 'of', 'five',
'minutes—see', ',', 'for', 'example', ',', 'Shieber', '(', '1994', ')', '.', 'It', 'was', 'widely', 'conceded',
'on', 'all', 'sides', 'that', 'the', 'programs', 'entered', 'in', 'the', 'Loebner', 'Prize', 'Competition',
'were', 'designed', 'solely', 'with', 'the', 'aim', 'of', 'winning', 'the', 'minor', 'prize', 'of', 'best',
'competitor', 'for', 'the', 'year', ',', 'with', 'no', 'thought', 'that', 'the', 'embodied', 'strategies',
'would', 'actually', 'yield', 'something', 'capable', 'of', 'passing', 'the', 'Turing', 'Test', '.', 'Midway',
'through', 'the', 'second', 'decade', 'of', 'the', 'twenty-first', 'century', ',', 'little', 'has', 'changed',
'.', '(', 'See', ',', 'for', 'example', ',', 'Floridi', '2008', '.', ')', 'True', 'enough', ',', 'in', '2014',
',', 'claims', 'emerged', 'that', ',', 'because', 'the', 'computer', 'program', 'Eugene', 'Goostman', 'had',
'fooled', '33', '%', 'of', 'judges', 'in', 'the', 'Turing', 'Test', '2014', 'competition', ',', 'it', 'had',
'“passed', 'the', 'Turing', 'Test”', '.', 'But', 'there', 'have', 'been', 'other', 'one-off']
```

Hacia el uso de métodos híbridos (1)

Como hemos visto, una forma de hacer análisis en documentos es empleando métodos de carácter simbólico, los cuales son ayudan a desarrollar diferentes tareas en varios niveles.

Hasta el momento, hemos visto dos clase de métodos: el diseño e implementación de autómatas, y el uso de expresiones regulares.



Hacia el uso de métodos híbridos (2)

Ahora, otra vía paralela es usar métodos estadísticos para hacer análisis y predicciones en distintos niveles del lenguaje.

4 Confusion Matrix

```
>>> reference = 'This is the reference data.  Testing 123.  aoaeoeoe'
>>> test =      'Thos iz_the rifirenci data.  Testeng 123.  aoaeoeoe'
>>> print ConfusionMatrix(reference, test)
  |   . 1 2 3 T _ a c d e f g h i n o r s t z |
--+-----------------------------------------+
  |<8>. . . . . 1 . . . . . . . . . . . . . |
. | .<2>. . . . . . . . . . . . . . . . . . |
1 | . .<1>. . . . . . . . . . . . . . . . . |
2 | . . .<1>. . . . . . . . . . . . . . . . |
3 | . . . .<1>. . . . . . . . . . . . . . . |
T | . . . . .<2>. . . . . . . . . . . . . . |
_ | . . . . . .<.>. . . . . . . . . . . . . |
a | . . . . . .<4>. . . . . . . . . . . . . |
c | . . . . . .<1>. . . . . . . . . . . . . |
d | . . . . . .<1>. . . . . . . . . . . . . |
e | . . . . . .<6>. . . 3 . . . . . . . . . |
f | . . . . . .<1>. . . . . . . . . . . . . |
g | . . . . . .<1>. . . . . . . . . . . . . |
h | . . . . . .<2>. . . . . . . . . . . . . |
i | . . . . . 1 . . .<1>. 1 . . . . . . . . |
n | . . . . . .<2>. . . . . . . . . . . . . |
o | . . . . . .<3>. . . . . . . . . . . . . |
r | . . . . . .<2>. . . . . . . . . . . . . |
s | . . . . . .<2>. 1 . . . . . . . . . . . |
t | . . . . . .<3>. . . . . . . . . . . . . |
z | . . . . . .<.>. . . . . . . . . . . . . |
--+-----------------------------------------+
(row = reference; col = test)
```

PLN trabaja mucho con métodos híbidos, ya que son muy poderosos porque justo combinan aspectos simbólicos y estadísticos.

Hacia el uso de métodos híbridos (3)

El uso de estadísticas no es algo exclusivo al PLN. Los lingüistas los hemos empleado por lo menos desde el S. XIX. Algunos de los *Junggrammatiker* (o neogramáticos) ya pensaban en sistemas combinatorios que ayudaban *a* explicar cambios lingüísticos entre las lenguas indoeuropeas.

Richard Montague

Alrededor de los 50 y los 60, un filosofo matemático estadounidense, Richard Montague (1930-1971) propuso algunos trabajos de lógica aplicados al ingles, p. e., analizo el comportamiento semántico de los cuantificadores.

Montague fue quizá el primero en plantear el uso de teoría de conjuntos y lógica modal para explicar la sintaxis y la semántica de las lenguas naturales.

Un autómata que sabe agrupar (1)

Lo que viene es un ejercicio. Si seguimos a Montague, suponemos viable describir un comportamiento sintáctico usando una lógica de conjuntos. Si esto es posible, incluso podemos modelar un autómata capaz de entender y reproducir este comportamiento.

Autómata para analizar frases nominales

Paso 1

i. En inglés, una frase nominal sigue una secuencia Artículo + Adjetivo + Nombre: *The White House*, *The New English Grammar*, *A big fish, those intelligent students…*

ii. En español, una frase nominal sigue una secuencia de Artículo + (Adjetivo) + Nombre + (Adjetivo), esto es, el adjetivo puede ir antes o después del nombre: *la prensa mexicana, el gran robo, mi nueva computadora negra…*

Un autómata que sabe agrupar (2)

Para que nuestro autómata entienda como se construye una buena frase nominal en inglés y en español, necesitamos buenos y malos ejemplos:

1. **En inglés:** An important theory [es correcta] / Book this expensive [es incorrecta]

1. **En español:** Un libro caro [es correcta]/ un buen libro [es correcta]/ un buen libro caro [¿suena bien?]/ Caro libro buen un [es incorrecta]

Un autómata que sabe agrupar (3)

Paso 2 Evaluando buenos y malos ejemplos, establecemos reglas de construcción.

Regla de construcción (RC)

1. Inglés: Art + (Adj*) + N
2. Español: Art + (Adj*) + N (Adj*)

Operadores (Op)

1. + = "Seguido de"
2. (…) = "Una palabra que puede aparecer o no"
3. * = "Se repite más de una vez"

Listas de palabras (LP)

1. Inglés: {The, a, this, dog, place, friend, brave, nice, intelligent}
2. Español: {El, un, ese, perro, lugar, amigo, fiero, buen, inteligente}

Un autómata que sabe agrupar (4)

Obviamente, conforme vamos estudiando nuestras reglas, las podemos hacer más finas.

Gramática de la frase nominal en español (GFNE)

GFNI = {RC & LP}

RC = {RC1 / RC2}

RC1 = ART + N + (ADJ*)

RC2 = ART + (ADJ*) + N

LP= {El, un, ese, perro, lugar, amigo, fiero, buen, inteligente}

Un autómata que sabe agrupar (5)

Paso 3

Segun mis reglas, ¿qué frases son correctas y/o incorrectas?

FN1 = El perro fiero

FN2 = Un perro fiero

FN3 = Ese perro fiero

FN4 = El buen perro

FN5 = El buen perro fiero

FN6 = El fiero buen perro

FN7 = El fiero buen inteligente perro

FN8 = El perro buen inteligente fiero (¿?)

FN9 = El perro amigo buen inteligente fiero (¿?)

FN10 = El perro amigo lugar buen inteligente fiero

Un autómata que sabe agrupar (6)

Supongamos que queremos analizar una lengua X, y queremos saber si se comporta como el inglés o el español.

Fase 1: Lista de palabras:

1. Der = “el”
2. Das = “el”
3. Mein = “Mi”
4. Hund = “perro/can”
5. Freund = “amigo”
6. Buch = “libro”
7. Gross = “gran(-de)”

Fase 2: Glosa:

1. Der = Art. Mas.
2. Das = Art. Neu. (Neutrum)
3. Mein = Art/Adj.
4. Hund = N.
5. Freund = N.
6. Buch = N.
7. Gross = Adj.

Un autómata que sabe agrupar (7)

Partiendo de las reglas que tiene nuestro autómata, supongamos que generamos algunas frases, y luego las evaluamos pidiéndole a un hablante de la *Lengua X* que nos diga si están bien o mal hechas.

Fase 3: producimos las siguientes oraciones.

FN1 = Der Hund
FN2 = Das Hund
FN3 = Mein Hund
FN4 = Der Mein Hund
FN5 = Der Gross Hund
FN6 = Der Hund Gross
FN7 = Der Mein Gross Hund
FN8 = Der Hund Mein Gross
FN9 = Der Mein Hund Gross
FN10 = Der Gross Hund Mein

Un autómata que sabe agrupar (8)

El Hablante de la *Lengua X* nos entrega su evaluación:

FN1 = Der Hund [Correcto]

FN2 = Das Hund [Incorrecto: no es lo mismo Masculino que Neutro]

FN3 = Mein Hund [Correcto. *¿Aplica para Masculino y Neutro?*]

FN4 = Der Mein Hund [Incorrecto: *Mein* no puede aparecer con *Der*]

FN5 = Der Gross Hund [Correcto: *¿la secuencia es Art + Adj* + N?*]

FN6 = Der Hund Gross [Incorrecto: la secuencia no es *Art + N + Adj**]

FN7 = Der Mein Gross Hund [Incorrecto: ¿entonces no se vale *Art + Adj* + N?*]

FN8 = Der Hund Mein Gross [Incorrecto: ¿no se vale *Art + N + Adj* ?*]

FN9 = Der Mein Hund Gross [Incorrecto: ¿no vale tampoco *Art + Adj* + N + Adj*?*]

FN10 = Der Gross Hund **Mein** [***Etwas*!:** ¿no es lo mismo que FN9*]*

Un autómata que sabe agrupar (9)

Hagamos lo mismo con el catalán. Tenemos algunas frases

Lista de palabras:

1. El = “el”
2. L’ = “el”
3. Meu = “Mi”
4. Gos = “perro”
5. Amic = “amigo”
6. Libre = “libro”
7. Vell = “viejo”

Glosa:

1. El = Art. Mas.
2. L’ = Art. Mas. (frente a vocal)
3. Meu = Art/Adj.
4. Gos = N.
5. Amic = N.
6. Libre = N.
7. Vell = Adj.

Un autómata que sabe agrupar (10)

Podemos generar algunas frases

FN1 = El gos

FN2 = El gos vell

FN3 = El vell gos

FN4 = El meu gos vell

FN5 = El meu vell gos

FN6 = El gos meu vell

FN7 = L'amic vell

FN8 = Le vell amic

FN9 = Le meu vell amic

FN10 = Le meu amic vell

Un autómata que sabe agrupar (11)

¿Funcionó?

FN1 = El gos [Correcto]

FN2 = El gos vell [Correcto]

FN3 = El vell gos [Correcto]

FN4 = El meu gos vell [¡Correcto!]

FN5 = El meu vell gos [¡Correcto!]

FN6 = El gos meu vell [¿Correcto?]

FN7 = L'amic vell [Correcto]

FN8 = Le vell amic [Correcto]

FN9 = Le meu vell amic [¡Correcto!]

FN10 = Le meu amic vell [¡Correcto!]

Un autómata que sabe agrupar (12)

Si pensamos como lógicos al estilo de Montague, lo que deseamos con estas pruebas es establecer una regla que nos permita modelar cómo se construyen las frases nominales en estas lenguas. Pensemos en que podemos fijar una regla “universal” para el inglés, el español, el alemán y el catalán:

$$\forall(x)\ FN(x) \rightarrow [(Art(y) \cap N(z)) \cap (ADJ(w)^{w0\text{-}wN})]$$

Un autómata que sabe agrupar (13)

Derivemos una regla exclusiva para el español, siguiendo la “regla universal”:

$$\exists (x)\ FN(x) \rightarrow [Art(y) \cup (ADJ(w)^{w0\text{-}wN}) \cap N(z) \cup (ADJ(w)^{w0\text{-}wN})]$$

La regla para el español se traduce como: *Existe una X que es frase nominal, la cual tiene un artículo Y, y/o cero o más adjetivos W, así como un nombre Z y/o cero o más adjetivos W.*

Un autómata que sabe agrupar (14)

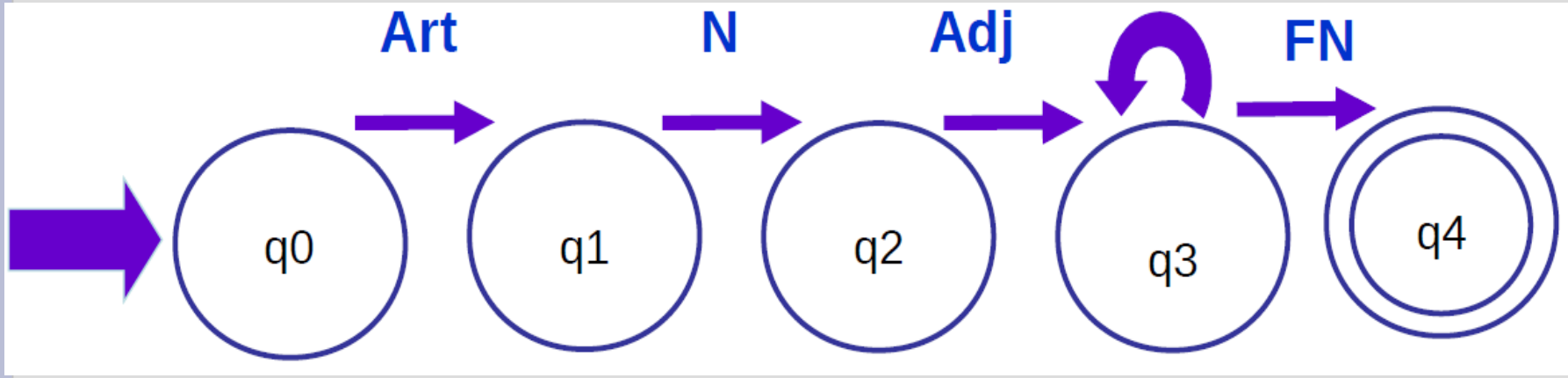
¿Y si hacemos lo mismo con el inglés? Parece que la regla sería:

$$\exists\,(x)\ FN(x) \rightarrow [Art(y) \cup (ADJ(w)^{w0\text{-}wN}) \cap N(z)]$$

La regla para el inglés equivale a decir: *Existe una X que es frase nominal, la cual tiene un artículo Y, y/o cero o más adjetivos W, junto con un nombre Z.*

Un autómata que sabe agrupar (15)

Empleando las proposiciones que hemos hecho, podemos diseñar un autómata que sea capaz de generar secuencias de frases nominales que correspondan con los patrones del inglés y el español. Tendríamos algo como:



Pregunta: este autómata por lo menos nos ayuda a resolver el caso del español, considerando que el adjetivo va en posición post-nominal. Si quisiéramos que este mismo autómata analizara el inglés, ¿qué tendríamos que modificar?

Gracias por su atención

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