

# Analizadores morfológicos aplicados al lenguaje natural, aplicaciones para búsqueda de información

Gunnar Wolf -

Desarrollador del Proyecto

Instituto de Investigaciones Económicas -

gwolf@

Debian

UNAM

# Aguas

- Voy a hablar en buena medida de cómo yo veo las cosas.
- A ratos no voy a usar los términos más aceptados en la literatura de la materia
- Esto incluye varias interpretaciones poco ortodoxas mías
- El tema me encanta. Eso es lo que cuenta, ¿no? ;-)

# Analizadores morfológicos aplicados al lenguaje natural, aplicaciones para búsqueda de información

## Contenido

1. Introducción
2. Ejemplo de una aplicación
3. TSearch2 en español a través de Snowball
4. ¿Y el verdadero análisis de lenguaje natural?

[http://www.gwolf.org/soft/an\\_morf\\_leng\\_nat/](http://www.gwolf.org/soft/an_morf_leng_nat/)

# 1. Introducción

¿De qué vamos a hablar?

---

- ¿Qué es un lenguaje? ¿Qué tipos de lenguaje hay?
- ¿Por qué puede interesarnos analizar el lenguaje natural?
- Ejemplo de una aplicación (simple pero válida) para la recuperación y análisis de información, aprovechando el análisis morfológico sobre lenguaje natural
- ¿Hasta dónde podemos llegar con el análisis?
- ¿Qué hay al respecto en la academia?

# 1. Introducción

¿Cuándo utilizamos el análisis de lenguajes?

---

- Prácticamente todo sistema de cómputo puede ser definido como un analizador de un lenguaje determinado
  - Toma datos de entrada
  - Procesa estos datos
  - Genera una salida
  - Posiblemente volviendo a la misma entrada, modificando su operación
  
- No en vano esto suena a la descripción de una máquina de Turing
  
- Hay diferentes tipos de lenguaje, que conllevan diferentes grados de complejidad. Podemos (casi) decir que cada una de las siguientes categorías engloba a sus predecesores.

# 1. Introducción

## Tipos de lenguajes: 1- Lenguajes descriptivos

---

- Describen una situación, un objeto, un estado, una configuración
- Realmente sencillos de manejar - ¿A quién se le ocurre realizar un estudio sobre de ellos?
  - ¡Los hay! Buscan una mayor simplicidad, legibilidad para humano y para computadora, etc.
- ¿Herramientas para su análisis?
  - Expresiones regulares / PCRE
  - Analizadores léxicos simples
  - Incluso analizadores gramaticales simples
- ¿Qué entra en esta clasificación?
  - Archivos de configuración
  - Datos generados (manual o automáticamente) expresamente para su interpretación
- Ejemplos
  - Archivos de configuración (planos, con jerarquía, etc.)
  - HTML / XML / YAML
  - Estructuras de datos simples (.DBF, .CSV, etc.)

# 1. Introducción

## Tipos de lenguajes: 2- Protocolos

---

- Hechos para permitir la comunicación entre (procesos|sistemas|computadoras) diferentes
  - Dentro de un universo de posibilidades de interacción claramente acotado
  
- No basta analizar un pedazo estático de información, hay que analizar una charla entre dos partes
  
- La estructura más adecuada para representar esta conversación es un autómata de estado finito
  - Una serie de estados, y eventos que ocasionan transiciones entre ellos
  - Estados iniciales, estados terminales
  
- ¿Qué entra en esta clasificación?
  - Prácticamente todos los protocolos de uso común en Internet (RFC)
  - Buena parte de los esquemas de comunicación entre procesos (IPC, RPC)

# 1. Introducción

## Tipos de lenguajes: 3- Lenguajes formales

---

- Hechos para permitir la descripción de un procedimiento (un algoritmo, un programa) o una relación compleja
- Toda construcción en un lenguaje formal se ciñe a una gramática determinada
- La información escrita en un lenguaje formal es expresable en términos de un árbol de análisis gramatical
- En caso de haber posibilidad de ambigüedad, debe haber una forma clara de resolverla
- La definición del lenguaje puede establecer ciertos supuestos, pero más allá de ellos, todo debe ser explícitamente indicado
- ¿Aplicaciones más comunes?
  - Compiladores
  - Procesadores de texto

# 1. Introducción

## Tipos de lenguajes: 4- Lenguajes naturales

---

- Los lenguajes empleados por los seres humanos para su comunicación
- Si bien todo lenguaje natural tiene un conjunto de reglas básicas, hay una cantidad espeluznante de excepciones
- Estamos acostumbrados a resolver ambigüedades
- Podemos razonar incluso con ausencia de información importante relativa al tema
  - Es imposible razonar con ausencia absoluta de información externa
  - Emplear lenguaje natural presupone un universo de conocimiento circundante

# Analizadores morfológicos aplicados al lenguaje natural, aplicaciones para búsqueda de información

## Contenido

1. Introducción
2. Ejemplo de una aplicación
3. TSearch2 en español a través de Snowball
4. ¿Y el verdadero análisis de lenguaje natural?

[http://www.gwolf.org/soft/an\\_morf\\_leng\\_nat/](http://www.gwolf.org/soft/an_morf_leng_nat/)

## 2. Ejemplo de una aplicación

Planteamiento de la problemática

---

# Cuerpo Académico Historia del Presente

<http://anuario.upn.mx/>

- Base de datos de artículos publicados en periódicos nacionales
- 50,264 artículos (al 15/2/2004), creciendo a ritmo de ~2500 mensuales
- Los artículos miden en promedio 3Kb, con máximos de hasta 70Kb
- La base de datos está en PostgreSQL, obviamente ;-)
- Los artículos se catalogan según una serie de criterios/categorías, pero la interfaz pública permite únicamente especificar fecha, diario, estado o texto a buscar

## 2. Ejemplo de una aplicación

### Planteamiento de la problemática: Velocidad

---

- ¿Qué pasa cuando una persona sólo busca una palabra, sin ninguna restricción?

```
EXPLAIN ANALYZE SELECT id from articulo where nota like '% analisis%';  
QUERY PLAN
```

```
-----  
Seq Scan on articulo (cost=0.00..19121.67 rows=1 width=4) (actual time=3833.432..41438.090 rows=2 loops=1)  
  Filter: (nota ~~ '% analisis% '::text)  
Total runtime: 41438.198 ms  
(3 rows)
```

- PostgreSQL no provee un mecanismo interno para indexar campos de texto completo (FTI - Full Text Index)
- Con suficiente memoria, para las búsquedas inmediatas subsecuentes el tiempo de consulta se reduce hasta al 5% (pues la base completa queda en cache)
  - No es escalable - Tarde o temprano la BD será más grande que la memoria
  - El uso normal del programa no implica búsquedas demasiado frecuentes
  - No es elegante

## 2. Ejemplo de una aplicación

### Planteamiento de la problemática: Inexactitud (1)

---

- Diferencia entre mayúsculas y minúsculas, acentos, etc.
  - 6190 artículos mencionan "Educación"
  - 5626 artículos mencionan "educación"
  - 104 artículos mencionan "EDUCACIÓN"
  - 147 artículos mencionan "educacion"
  - 37 artículos mencionan "Educacion"
  - 9 artículos mencionan "EDUCACIÓN"
  - 8614 artículos totales (una palabra aparece de diferentes maneras en el mismo)
  - ...Seguro hay términos con mayor variabilidad.
  
- Este problema se aminora con el uso del operador ~\* en vez de LIKE
  - La comparación con regex es algo más lenta que la textual (
    - ▶ Aunque comparando con el tiempo que toma el acceso al disco, puede ser irrelevante
  - El problema sigue sin resolverse del todo (8416 para "educación", 280 para "educacion"). No podemos asumir que todo usuario del sistema piense en ello

## 2. Ejemplo de una aplicación

### Planteamiento de la problemática: Inexactitud (2)

---

- Los términos relacionados deberían aparecer también como resultados
  - 13163 artículos para "educativo"
  - 1179 artículos para "educado"
  - 1047 artículos para "educador"
  - 739 artículos para "educar"
  - 665 artículos para "educando"
  - 73 artículos para "eduque"
  
- ...Y no estamos tomando en cuenta sinónimos, parónimos, etc.
  - 9069 artículos para "conocimiento" mas sus variaciones (2538 no mencionan "educación"+variantes)
  - 2007 para "enseñanza" (574 no mencionan "educación")
  - 1864 para "instrucción" (852 no mencionan "educación")

## 2. Ejemplo de una aplicación

### Buscando la solución

---

- La problemática que busqué resolver en un principio fue sólo la de la velocidad
- Por un sistema previo en el que había trabajado, conocía un proyecto en Postgres que implementaba indexado sobre texto completo: FTI
  - Pero no es ya considerado la mejor opción por los desarrolladores de Postgres
  - Requiere la construcción de tablas adicionales para cada punto donde se emplea
  - Tiene una sintaxis poco natural, requiere estar amarrando tablas por OIDs
  - Carece de un mecanismo que ordene por relevancia
- El proyecto TSearch2 es desde la versión 7.3 de Postgres suficientemente maduro para ser incluido como parte del **contrib** oficial

## 2. Ejemplo de una aplicación

### Detalles de la implementación de TSearch2

---

- TSearch2 basa su indexación en el Árbol de Búsqueda Generalizado (GiST), desarrollado por Teodor Sigaev y Oleg Bartunov
  - Generaliza los mecanismos de indexación para presentar una interfaz adecuada para diferentes tipos de comparaciones/criterios
  
- Crea siete tipos de datos adicionales. Los principales que manipularemos directamente son:
  - **tsvector**: Conjunto de lexemas únicos junto con su posición en el documento, con una estructura optimizada para acceso y búsqueda rápidos. Todo documento es convertido en su tsvector, y éste es indexado.
  - **tsquery**: Combinación booleana de lexemas, utilizada para realizar las búsquedas sobre el índice de tsvector
  
- Crea cuatro tablas para regular el comportamiento general
  - **pg\_ts\_dict**: Definiciones de los diccionarios a emplear
  - **pg\_ts\_parser**: Referencia (por OID) a los analizadores morfológicos disponibles
  - **pg\_ts\_cfg**: Qué reglas utilizar con qué locales
  - **pg\_ts\_cfgmap**: Qué diccionario aplicar a cada tipo de datos dependiendo del juego de reglas seleccionado

## 2. Ejemplo de una aplicación

### Echando a andar TSearch2 (1)

---

- Requisitos generales: PostgreSQL 7.3 o superior, postgresql-contrib
- Crear los tipos de datos, tablas y funciones de TSearch2 en nuestra BD  
\$ psql hist\_pres  
hist\_pres=# \i <path>/tsearch2.sql  
En el caso de Debian, tsearch2.sql está en /usr/share/postgresql/contrib/

- Verificamos que la funcionalidad de TSearch2 exista

```
hist_pres=# SELECT 'Esta es una prueba'::tsvector;  
                tsvector
```

```
-----
```

```
'es' 'una' 'Esta' 'prueba'
```

```
(1 row)
```

```
hist_pres=# SELECT to_tsvector('simple','Esta es una prueba,  
una prueba más');
```

```
                to_tsvector
```

```
-----
```

```
'es':2 'más':7 'una':3,5 'esta':1 'prueba':4,6
```

```
(1 row)
```

## 2. Ejemplo de una aplicación

### Echando a andar TSearch2 (2)

---

- Creamos o modificamos nuestra tabla para que tenga un campo para el índice y generamos los índices sobre los datos ya existentes (opero en este caso sólo sobre una fracción de los datos)

```
hist_pres=# ALTER TABLE articulo ADD column ts2 tsvector;  
ALTER TABLE  
hist_pres=# UPDATE articulo SET ts2=to_tsvector('simple',  
        coalesce(title,'') || ' ' || coalesce(nota,'');  
UPDATE 1000  
hist_pres=# VACUUM FULL ANALYZE;  
VACUUM
```

- Creamos el índice sobre este registro, y el trigger para mantenerlo actualizado

```
hist_pres=# CREATE INDEX ts2_idx ON articulo USING gist(ts2);  
CREATE INDEX  
hist_pres=# VACUUM FULL ANALYZE;  
VACUUM  
hist_pres=# CREATE TRIGGER ts2_update BEFORE UPDATE OR INSERT  
        ON articulo FOR EACH ROW EXECUTE PROCEDURE  
        tsearch2(ts2, titulo, nota);  
CREATE TRIGGER
```

## 2. Ejemplo de una aplicación

### Probando TSearch2

---

- Hacemos nuestras primeras búsquedas

```
SELECT id, rank(ts2,'ciencia | tecnología'::tsquery) AS r, headline('simple',titulo,q) FROM
articulo, to_tsquery('simple','ciencia | tecnología') as q WHERE ts2 @@ q ORDER BY r DESC LIMIT 5;
```

id	r	headline
42094	0.998545	En México, la <b>ciencia</b> tiene dos obstáculos: desinterés y desconocimiento
9666	0.94185	Abandonadas, las fuerzas básicas de la <b>ciencia</b>
26645	0.935389	Educación, <b>ciencia</b> y comportamiento ético
48347	0.920234	En su futuro, México será un país sin médicos ni matemáticos
47962	0.864915	Impacta en los estados recorte a <b>ciencia</b>

(5 rows)

- Funciona correctamente y hasta bonito... ¡Pero la búsqueda es más lenta que usando un LIKE!
- Además, sigue reportándonos por separado los artículos de tecnología y los detecnologia
- ...Hora de echarse un clavadito a la documentación

# Analizadores morfológicos aplicados al lenguaje natural, aplicaciones para búsqueda de información

## Contenido

1. Introducción
2. Ejemplo de una aplicación
3. TSearch2 en español a través de Snowball
4. ¿Y el verdadero análisis de lenguaje natural?

[http://www.gwolf.org/soft/an\\_morf\\_leng\\_nat/](http://www.gwolf.org/soft/an_morf_leng_nat/)

### 3. TSearch2 en español a través de Snowball

#### Los diferentes diccionarios

---

- Pueden haber notado que en algunas consultas menciono

- `SELECT to_tsvector('simple','Esta es una prueba, una prueba más');`
- `UPDATE articulo SET ts2=to_tsvector('simple', coalesce(title,'') || ' ' || coalesce(nota,''));`

- Con la distribución de TSearch2 vienen cinco diccionarios:

```
hist_pres=# select dict_name, dict_comment from pg_ts_dict ;
 dict_name | dict_comment
-----+-----
 simple   | Simple example of dictionary.
 en_stem  | English Stemmer. Snowball.
 ru_stem  | Russian Stemmer. Snowball.
 ispell_template | ISpell interface. Must have .dict and .aff files
 synonym  | Example of synonym dictionary
(5 rows)
```

- Usando Gendict podemos definir nuevos diccionarios basados en en alizadores morfológicos

- Un diccionario no es simplemente una lista de palabras - Un diccionario recibe palabras textuales y entrega lexemas, que son típicamente una forma reducida de esta palabra.

### 3. TSearch2 en español a través de Snowball

#### Tipos de diccionario

---

Los diccionarios predefinidos que vienen con TSearch2 son:

- Simple: Diccionario nulo
- ispell\_template: Basados en listas de palabras
  - Usa una interfaz hacia ispell
  - No agrupa las palabras relacionadas
  - El análisis es mucho más rápido
- en\_stem, ru\_stem: Basados en un stemmer (analizador morfológico)
  - Conjunto de reglas no atadas a un vocabulario predefinido
  - Agrupan palabras relacionadas bajo un mismo concepto
  - Uso de una lista de stopwords
  - El análisis toma más tiempo
    - ▶ ...Pero se hace una sólo vez por registro
- synonym: Listas de sinónimos
  - Agrupan palabras cuyo significado está relacionado entre sí
  - La implementación por defecto es un ejemplo incompleto (?)

Aparte de estos, podemos utilizar diccionarios definidos por el usuario, la documentación para hacerlo es bastante completa

### 3. TSearch2 en español a través de Snowball

#### El proyecto Snowball

---

- Martin Porter diseñó en 1980 un algoritmo que, dada una palabra en inglés, entrega el lexema (esto es, el componente común a todas las conjugaciones, modos y casos de la palabra).
- El algoritmo original de Porter fue escrito en BCPL, pero ha sido traducido a los principales lenguajes en uso hoy en día (disponibles en el sitio Web de Porter, <http://www.tartarus.org/~martin/>)
- ¿Los problemas de el algoritmo de Porter, así como de otros algoritmos similares (p.ej. el de Kraaij, 1994, para el holandés)?
  - Hay muchas implementaciones erróneas (márgenes de error superiores al 10%)
  - Es difícil entender un algoritmo viendo el código que lo ejecuta
  - No todos los algoritmos pueden ser fácilmente embebidos
- Para resolver estos problemas, Porter escribió el lenguaje **Snowball**
  - El lenguaje busca ser muy simple de aprender
  - Permite expresar fácilmente las reglas léxicas de un lenguaje
  - Puede ser embebido en diversas aplicaciones
  - Su nombre es un tributo al lenguaje SNOBOL, un poderoso lenguaje de manipulación de cadenas ideado en Bell Labs a principios de los 60.

### 3. TSearch2 en español a través de Snowball

#### Utilizando Snowball desde TSearch2 (1)

---

El diccionario `en_stem` (utilizado desde la configuración `default`) utiliza el análisis del inglés de Snowball.

```
hist_pres=# SELECT to_tsvector('simple','We are stemming and testing for occurrence of common words');
               to_tsvector
```

```
-----
 'of':8 'we':1 'and':4 'are':2 'for':6 'words':10 'common':9 'testing':5 'stemming':3 'occurrence':7
(1 row)
```

```
hist_pres=# SELECT to_tsvector('default','We are stemming and testing for occurrence of common words');
               to_tsvector
```

```
-----
 'stem':3 'test':5 'word':10 'common':9 'occurr':7
(1 row)
```

- La longitud total del tsvector generado es de la mitad
- Las palabras fueron reducidas a su mínima forma única

### 3. TSearch2 en español a través de Snowball

#### Utilizando Snowball desde TSearch2 (2)

---

Para las funciones de búsqueda

- Creación de cadenas de búsqueda

```
hist_pres=# SELECT to_tsquery('default', 'query & optimization & ! (algorithmic | methodic)');
               to_tsquery
-----
'queri' & 'optim' & !( 'algorithm' | 'method' )
(1 row)
```

- Marcado de resultados

```
hist_pres=# SELECT headline('default','I am not testing this. You are welcome to run any tests it
defines',to_tsquery('default','test'));
               headline
-----
I am not <b>testing</b> this. You are welcome to run any <b>tests</b> it defines
(1 row)
```

### 3. TSearch2 en español a través de Snowball

#### Snowball en español desde TSearch2 (1)

---

- Bajamos la implementación en C del analizador Snowball (`stem.h` y `stem.c`) y la lista de stop-words (`stop.txt`) desde <http://snowball.tartarus.org/spanish/stemmer.html>
- Bajamos el código fuente y las dependencias de compilación de PostgreSQL
  - Incluso en el caso de que usemos un paquete de PostgreSQL definido por nuestra distribución, requerimos las fuentes para compilar este módulo
  - Requerimos configurar PostgreSQL y compilar TSearch2. Para un caso mínimo:

```
postgresql-7.4.7$ ./configure
(...)
postgresql-7.4.7$ cd contrib/tsearch2
postgresql-7.4.7/contrib/tsearch2$ make
(...)
```
- Copiamos `stem.h` y `stem.c` a `contrib/tsearch2/gendict`, y desde ese directorio:

```
(...)/tsearch2$ cd gendict;
(...)/tsearch2/gendict$ ./config.sh -n sp -s -p spanish -v
-c'Snowball stemmer for Spanish'
```

### 3. TSearch2 en español a través de Snowball

#### Snowball en español desde TSearch2 (2)

---

- Vamos al directorio que config.sh nos indica y compilamos

```
(...)/tsearch2/gendict$ cd ../../dict_sp  
(...)/dict_sp$ make
```

- La documentación de gendict sugiere hacer `make install`, pero esto no verifica cuál es el `PGLIB` - Yo sugiero encargarse de este paso a mano - En caso de haber cualquier variación, recurrir a `make install` + un poco de depuración manual

```
(...)/dict_sp$ su  
(...)/dict_sp# cp libdict_sp.so.0.0  
/usr/lib/postgresql/lib/dict_sp.so
```

- Creamos las llamadas a las funciones adecuadas desde nuestra base de datos usando el archivo `dict_sp.sql` generado:

```
/dict_sp$ psql hist_pres  
hist_pres=# \i dict_sp.sql
```

- Agregamos la referencia a la lista de stopwords en español

```
hist_pres=# UPDATE pg_ts_dict SET  
dict_initoption='/usr/share/postgresql/lib/spanish.stop' WHERE  
dict_name='sp');
```

### 3. TSearch2 en español a través de Snowball

#### Snowball en español desde TSearch2 (3)

---

- Verificamos que funcione el soporte al español

```
hist_pres=# SELECT lexize('sp','probando');
lexize
-----
 {prob}
(1 row)
```

- Indicamos a Postgres que use la lista de stopwords que bajamos

```
hist_pres=# UPDATE pg_ts_dict SET dict_initoption =
'/usr/share/postgresql/contrib/spanish.stop' WHERE dict_name =
'sp';
UPDATE 1
```

### 3. TSearch2 en español a través de Snowball

#### Snowball en español desde TSearch2 (4)

---

- Creamos un juego de reglas para el uso del español

```
hist_pres=# INSERT INTO pg_ts_cfg (ts_name, prs_name, locale) VALUES ('espanol','default', NULL);
```

- Aplicamos estas reglas para el tipo de lexemas que indiquemos

```
hist_pres=# INSERT INTO pg_ts_cfgmap (ts_name, tok_alias, dict_name) VALUES ('espanol','lword','{sp}');
```

```
hist_pres=# INSERT INTO pg_ts_cfgmap (ts_name, tok_alias, dict_name) VALUES ('espanol','lpart_word','{sp}');
```

```
hist_pres=# INSERT INTO pg_ts_cfgmap (ts_name, tok_alias, dict_name) VALUES ('espanol','lhword','{sp}');
```

```
hist_pres=# INSERT INTO pg_ts_cfgmap (ts_name, tok_alias, dict_name) VALUES ('espanol','lpart_hword','{sp}');
```

```
hist_pres=# INSERT INTO pg_ts_cfgmap (ts_name, tok_alias, dict_name) VALUES ('espanol','url','{simple}');
```

```
hist_pres=# INSERT INTO pg_ts_cfgmap (ts_name, tok_alias, dict_name) VALUES ('espanol','host','{simple}');
```

```
hist_pres=# INSERT INTO pg_ts_cfgmap (ts_name, tok_alias, dict_name) VALUES ('espanol','uint','{simple}');
```

```
hist_pres=# INSERT INTO pg_ts_cfgmap (ts_name, tok_alias, dict_name) VALUES ('espanol','int','{simple}');
```

```
hist_pres=# INSERT INTO pg_ts_cfgmap (ts_name, tok_alias, dict_name) VALUES ('espanol','float','{simple}');
```

- Seleccionamos trabajar en esta sesión con la configuración especificada

```
hist_pres=# SELECT set_curcfg('espanol');
```

### 3. TSearch2 en español a través de Snowball

#### Snowball en español desde TSearch2 (5)

---

Hay varios tipos de lexema para Tsearch2, utilizados por su parser:

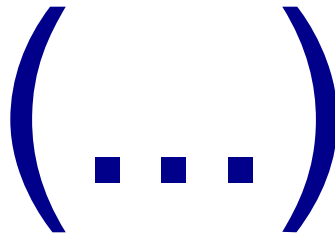
```
ts2test=# SELECT * from token_type();
```

tokid	alias	descr
1	lword	Latin word
2	nlword	Non-latin word
3	word	Word
4	email	Email
5	url	URL
6	host	Host
7	sfloat	Scientific notation
8	version	VERSION
9	part_hword	Part of hyphenated word
10	nlpart_hword	Non-latin part of hyphenated word
11	lpart_hword	Latin part of hyphenated word
12	blank	Space symbols
13	tag	HTML Tag
14	http	HTTP head
15	hword	Hyphenated word
16	lhword	Latin hyphenated word
17	nlhword	Non-latin hyphenated word
18	uri	URI
19	file	File or path name
20	float	Decimal notation
21	int	Signed integer
22	uint	Unsigned integer
23	entity	HTML Entity

### 3. TSearch2 en español a través de Snowball

Snowball en español desde TSearch2 (6)

---



Hablando con la verdad, no puedo continuar exponiendo respecto a TSearch2 en español

- El desarrollo de la aplicación de Historia del Presente me exigió dar prioridad a terminar con otras partes de la funcionalidad, relegando lo bonito y divertido para más tarde
- Hay que pensar cómo manejar la interfaz al usuario, dado que el comportamiento de una búsqueda tras pasar por el analizador morfológico es diferente del de una búsqueda tradicional

...Continuemos, pues, viendo cosas bonitas y divertidas ;-)

# Analizadores morfológicos aplicados al lenguaje natural, aplicaciones para búsqueda de información

## Contenido

1. Introducción
2. Ejemplo de una aplicación
3. TSearch2 en español a través de Snowball
4. ¿Y el verdadero análisis de lenguaje natural?

[http://www.gwolf.org/soft/an\\_morf\\_leng\\_nat/](http://www.gwolf.org/soft/an_morf_leng_nat/)

## 4. ¿Y el verdadero análisis de lenguaje natural?

### Aplicaciones del procesamiento del lenguaje natural

---

Hay mucha gente estudiando diversos aspectos del análisis de lenguaje natural. Las principales aplicaciones sugeridas son:

- Traducción automática
  - Convertir un lenguaje natural en su representación simbólica, y convertir a ésta de nuevo a lenguaje natural
  - Para traducir no basta con tener una lista de palabras, es necesario entender lo que se quiere expresar
    - ▶ The spirit is strong but the flesh is weak => The vodka is strong but the flesh is rotten
  
- Minería de datos
  - Procesar grandes cantidades de información para inferir información adicional
  
- Interfaces hombre-máquina no textuales
  - Reconocimiento inteligente de voz

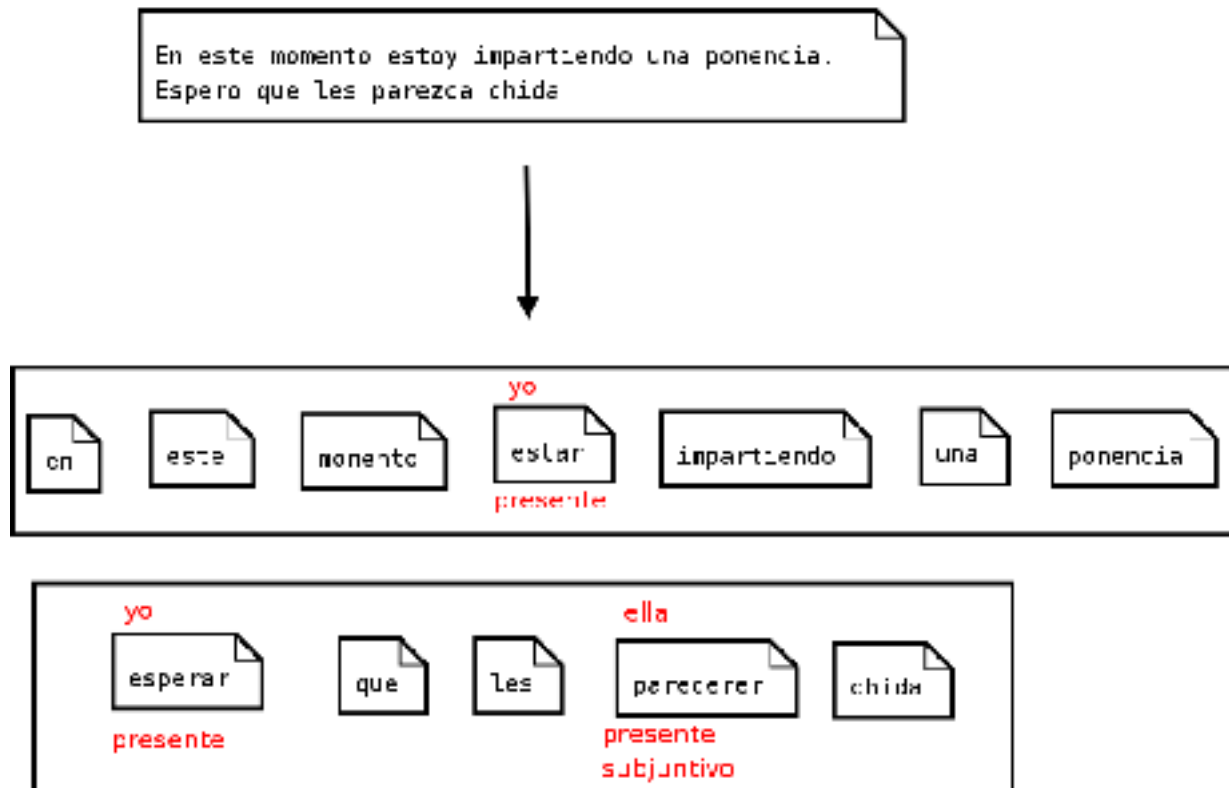
## 4. ¿Y el verdadero análisis de lenguaje natural?

Puntos en común a (casi) todas estas aplicaciones

---

- La aplicación que vimos cubre únicamente el **análisis morfológico** - y no a profundidad
  - Separa un flujo de texto en una serie de tokens aptos para ser procesados por el sistema
  - Hay pérdida de información: análisis gramatical/estructural (características de la palabra / rol de la palabra dentro de la oración), por lo que carece de utilidad para comprender el texto en cuestión

### Analisis lexico



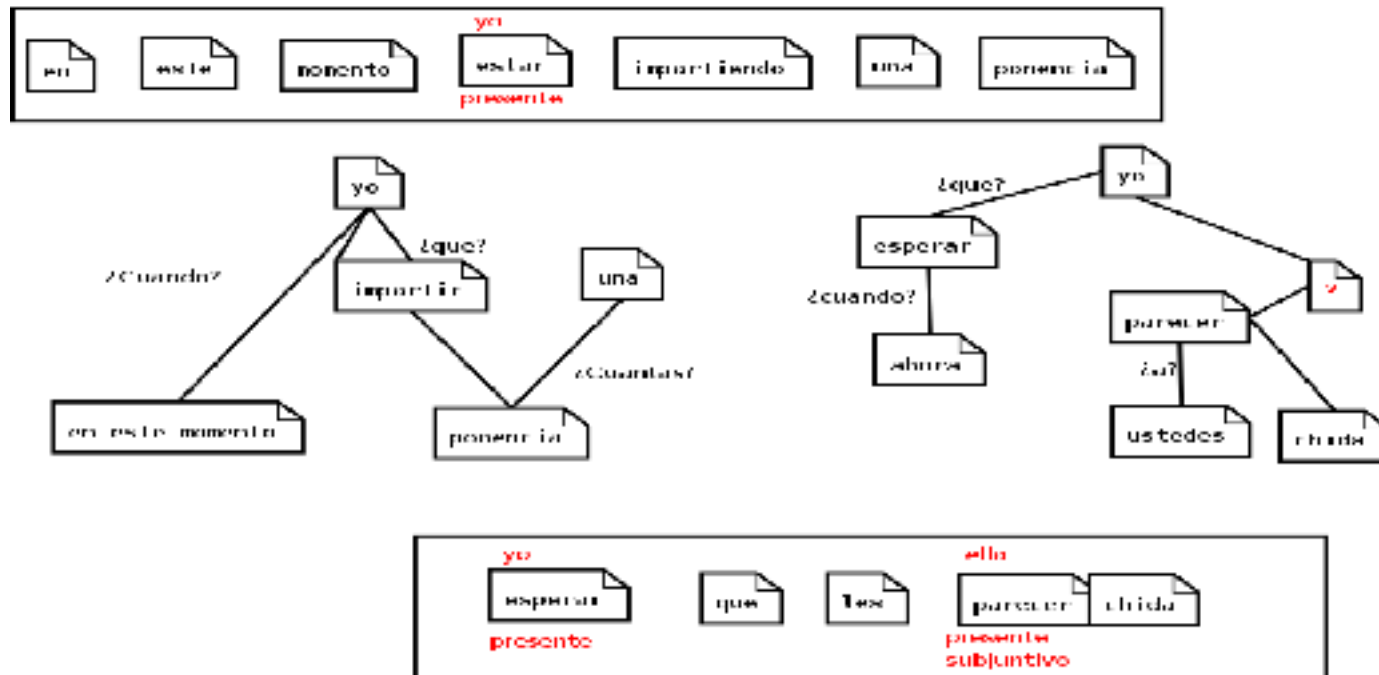
## 4. ¿Y el verdadero análisis de lenguaje natural?

Puntos en común a (casi) todas estas aplicaciones

---

- Más allá del análisis morfológico y léxico, todo análisis requerirá la construcción de un árbol de conceptos - El **análisis gramatical**: Une los diferentes tokens que nos arroja el análisis semántico según sus propiedades y su posición en la frase

### Analisis gramatical

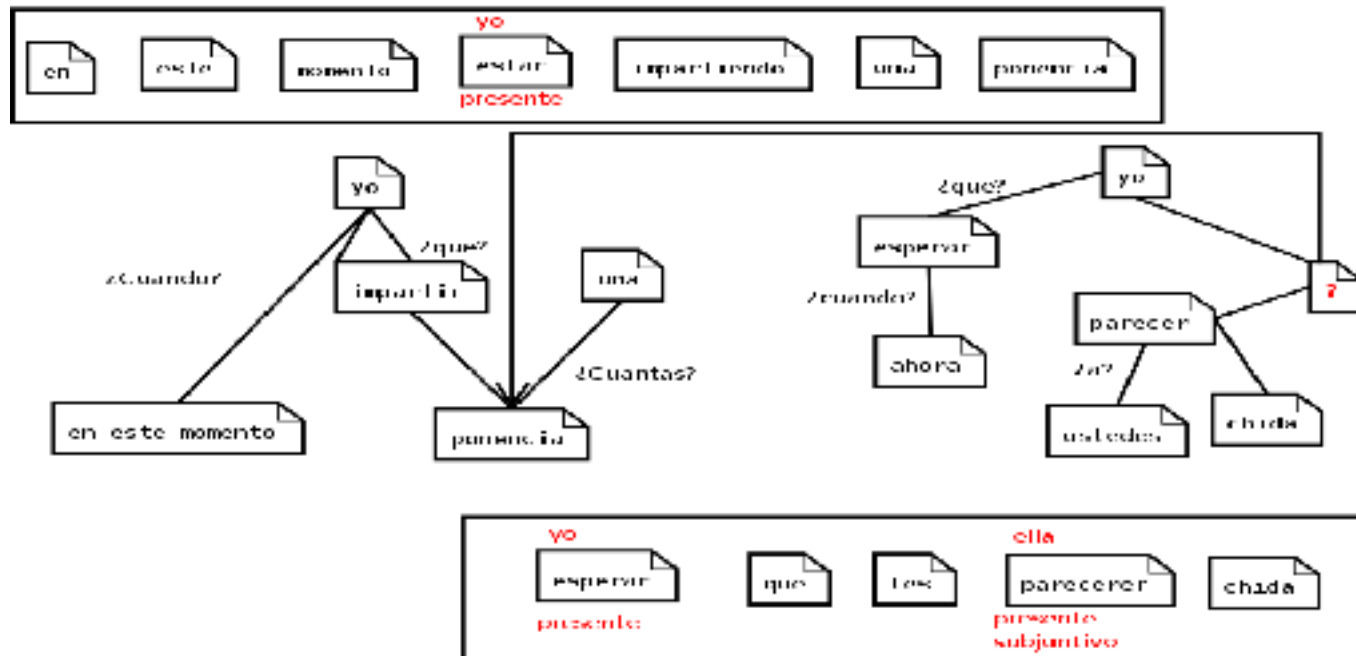


## 4. ¿Y el verdadero análisis de lenguaje natural?

Puntos en común a (casi) todas estas aplicaciones

- Casi siempre hace un ligado entre los diferentes árboles para completar los huecos que se refieren a otros nodos del árbol - Y aquí es donde comenzamos a alejarnos del diseño tradicional de compiladores de lenguajes formales

### Analisis gramatical (2)



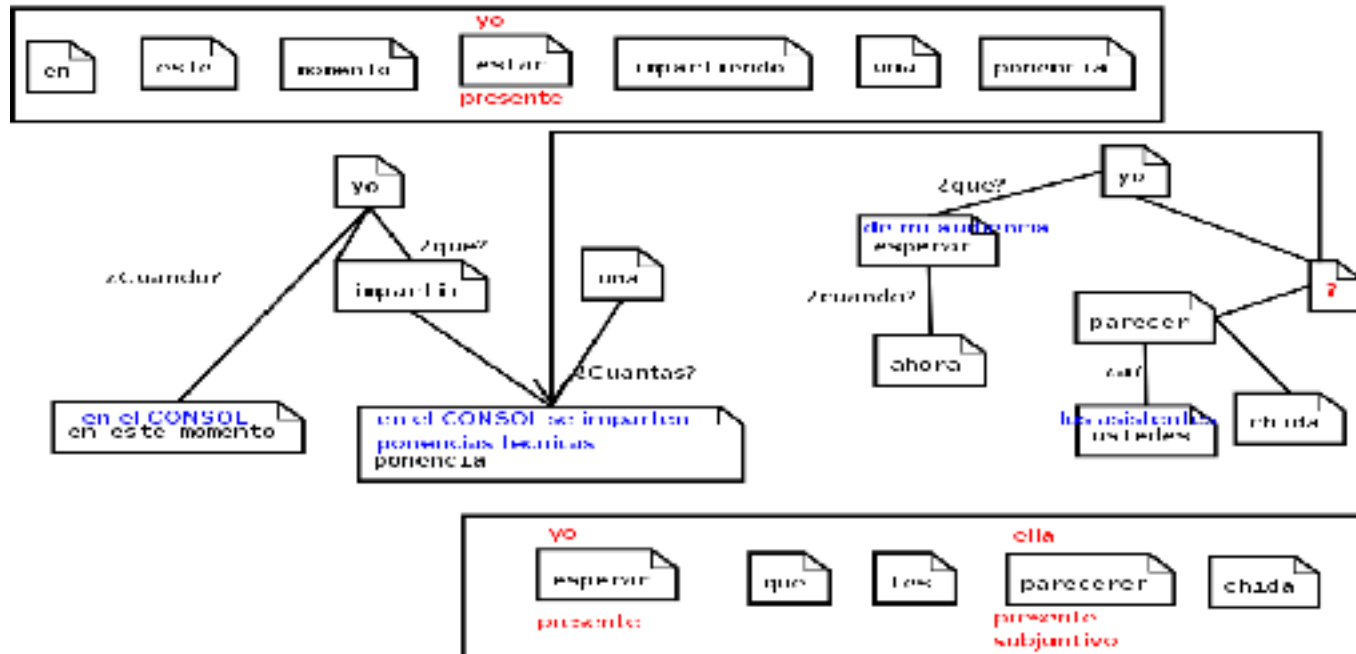
## 4. ¿Y el verdadero análisis de lenguaje natural?

Puntos en común a (casi) todas estas aplicaciones

---

- Muy frecuentemente le agregamos el conocimiento relevante para nuestra tarea que tengamos acerca del universo circundante

### Analisis gramatical (3)



## 4. ¿Y el verdadero análisis de lenguaje natural?

Puntos en común a (casi) todas estas aplicaciones

---

- El análisis pragmático nos ayuda a reducir ambigüedades
  - ¿Qué es lo que quiere decir?
  - ¿Qué tan probable es que quiera decir eso?

### **chido:**

- 1. Divertido, entretenido.
- 2. Bien hecho (en contexto familiar)
- 3. Naco es chido

En el CONSOL (un congreso académico de gran prestigio) probablemente yo quiere que audiencia considere a su ponencia sea vista como divertida, entretenida, aunque también puede aplicar bien hecho. Es muy poco probable que quiera decir naco.

- El análisis pragmático se hace tras procesar el documento completo
  - Probablemente dando mayor importancia al contexto local, determinado por párrafos

# ¿Dudas?

[gwolf@gwolf.org](mailto:gwolf@gwolf.org)

[http://www.gwolf.org/soft/an\\_morf\\_leng\\_nat/](http://www.gwolf.org/soft/an_morf_leng_nat/)

## TSearch2

- Implementing Full Text Indexing with PostgreSQL, Joshua D. Drake (Agosto 2004)  
<http://www.devx.com/opensource/Article/21674/1954?pf=true>
- GiST: A Generalized Search Tree for Secondary Storage, Teodor Sigaev y Oleg Bartunov,  
<http://gist.cs.berkeley.edu/gist1.html>
- Tsearch2 - Introduction, Andrew J. Kopciuch,  
<http://www.sai.msu.su/~megera/postgres/gist/tsearch/V2/docs/tsearch-V2-intro.html>
- Gendict - Generate dictionary templates for TSearch2 module,  
<http://www.sai.msu.su/~megera/oddmuse/index.cgi/Gendict>
- A Walkthrough for Making A Custom Dictionaries for tsearch2, Ben Chobot (Marzo 2004),  
<http://www.sai.msu.su/~megera/postgres/gist/tsearch/V2/docs/custom-dict.html>
- The tsearch2 Guide, Brandon Craig Rhodes, June 2003,  
<http://www.rhodesmill.org/brandon/projects/tsearch2-guide.html>

# Bibliografía

---

## **Snowball**

- M.F. Porter Snowball: A language for stemming algorithms (Octubre 2001)  
<http://snowball.tartarus.org/texts/introduction.html>
- Porter, M.F., 1980, An algorithm for suffix stripping, Program, 14(3) :130-137.  
<http://www.tartarus.org/~martin/PorterStemmer/def.txt>
- Spanish stemming algorithm, <http://snowball.tartarus.org/spanish/stemmer.html>

## **Análisis de lenguaje natural**

- Alexander Gelbukh, Igor Bolshakov. Avances y Perspectivas de Procesamiento Automático de Lenguaje Natural. J. IPN Ciencia, Arte: Cultura, N 31 Vol. II, Mayo-Junio 2000, ISSN 1405-2822. IPN, Mexico, pp. 10-18.  
<http://main.cicling.org/gelbukh/CV/Publications/2000/IPN-Proc-Leng-Nat.htm>
- A. Gelbukh. Tendencias recientes en el procesamiento de lenguaje natural. Proc. SICOM-2002, Villahermosa, Tabasco, México, 2002.  
<http://main.cicling.org/gelbukh/CV/Publications/2002/SICOM-2002-Trends.htm>