|  |
| --- |
| Procesamiento de lenguaje natural |
| Práctica 1: Síntesis |
|  |
| Gabriel Vázquez Torres |
|  |

Procesamiento de lenguaje natural

Práctica 1: Síntesis

# Definiciones importantes

**Fonética y fonología**: conocimiento sobre sonidos lingüísticos.

**Morfología**: conocimiento de los componentes significativos de las palabras.

**Sintaxis**: conocimiento de las relaciones estructurales entre palabras.

**Semántica**: conocimiento del significado.

**Pragmática**: conocimiento de la relación de significado de los objetivos e intenciones del hablante.

**Discurso**: conocimiento sobre unidades lingüísticas más grandes que un solo enunciado.

# Introducción

A continuación, se hará una serie de resúmenes de varios capítulos del libro “Speech and Language Processing (2nd editon)”, específicamente de los capítulos 2,3,4 y 5 tratando, a la vez, de responder una serie de cuestiones ofrecidas en la práctica 1 de la asignatura “Procesamiento de Lenguaje Natural”.

# Resumen teórico de capítulo 2

## Introducción

A forma de introducción al tema, podemos imaginarnos lo siguiente:

* Podemos encontrarnos con un documento que tenga una palabra en plural (“números”) y en singular (“número”). Ahora queremos reemplazar ambas palabras por otra en particular. Realmente preferiríamos no tener que buscar cada palabra por separado sino tener una función que detectara que quieres cambiar una palabra y su plural (con “s” al final). O, tal vez, quieras encontrar en un documento de texto todas las cadenas que se parezcan a valores numéricos. En este caso, preferiríamos tener una función que detectara valores numéricos aunque sea con decimales (199, 25, 24.99, etc).

Esto hace referencia a lo que veremos en este capítulo:

* En este capítulo se introduce a **las expresiones regulares, la notación estándar de caracterización de secuencias de texto**. Las expresiones regulares se utilizan para especificar cadenas de texto en todo tipo de aplicaciones de procesamiento de texto y extracción de información.
* Después de introducirnos a las expresiones regulares veremos cómo las podemos implementar con el autómata de estados finitos. El autómata de estados finitos no es solo un dispositivo matemático usado para implementar expresiones regulares. Es una de las mejores herramientas en computación lingüística.
* Variaciones del autómata como los modelos ocultos de Markov los veremos en capítulos más adelante.

## Expresiones regulares

Las expresiones regulares se utilizan en casi todo tipo de lenguajes de programación, incluso en Microsoft Word y muchas características de ER (Expresiones regulares) se utilizan en las búsquedas Web. Además, son una buena herramienta para la informática y la lingüística.

**Una expresión regular (creada en primera instancia por Kleene (1956)) es una fórmula en un lenguaje especial que especifica clases simples de cadenas.** Una cadena es una secuencia de símbolos alfanuméricos.

Formalmente, una expresión regular es una notación algebraica para caracterizar un conjunto de cadenas. En consecuencia, pueden especificar búsquedas de cadenas tan bien como lo define un lenguaje formal.

Las expresiones regulares requieren de un patrón que nosotros queremos buscar y de un corpus de textos para buscar a través de él. La función de búsqueda de la expresión regular buscará a través del corpus, devolviendo todos los textos que coincidan con el patrón.

En un sistema de recuperación de la información como un motor de búsqueda WEB, los textos pueden contener documentos. En un procesador de textos, los textos pueden ser palabras individuales o líneas de un documento. Esto lo utilizaremos en el resto del capítulo como paradigma.

### Patrones básicos de expresiones regulares

La expresión regular más sencilla es una secuencia de caracteres simples. Por ejemplo, para buscar “woodchuck”, nosotros escribiremos /woodchuck/. La búsqueda puede consistir, en una palabra, grupo de palabras o un carácter.

Existen muchas formas de representar expresiones regulares y hay muchas técnicas para ello. Al ser un resumen no entraremos en detalle en esto ya que lo veremos en la práctica siguiente a través del código.

Por comentar alguno en especial, comentaremos los anclajes (Anchors), que son caracteres especiales que anclan una expresión en un lugar en particular de un string. Los Anchors más comunes son ^ y $. En la práctica siguiente lo veremos en el código.

## Autómata de estado finito

Tres definiciones a tener en cuenta en este punto:

1. Una expresión regular es una forma de describir el autómata de estado finito (FSA, de sus siglas en inglés). Ninguna expresión regular puede ser implementada como un autómata de estado finito (excepto las expresiones regulares que usan la característica de memoria). De igual forma, ningún autómata de estado finito puede ser descrito como una expresión regular.
2. Una expresión regular es una manera de caracterizar un tipo especial de lenguaje formal llamado “lenguaje regular”.
3. Una tercera forma de caracterizar un lenguaje regular es la “gramática regular”.

Ambas expresiones pueden ser usadas para describir a los lenguajes regulares.



Ilustración 1: Tres maneras equivalentes de describir un lenguaje regular.

### Uso de un FSA para reconocer el “habla de ovejas” (onomatopeyas)

En este punto se habla sobre la posibilidad de crear FSA que modele una expresión regular para un caso en concreto. En el caso de la onomatopeya “baa!”, podemos crear la expresión regular /baa+!/ para reconocer la extensión de la onomatopeya, e.g., “baaaaaa!”.

El “autómata” (FSA) reconoce el conjunto de Strings de la misma manera en la que actúa una expresión regular.

Se puede representar al autómata como un grafo dirigido: un conjunto finito de vértices o nodos. En el caso de la palabra “baa”, el autómata tendría cinco estados. Se puede representar de la siguiente manera:



Ilustración 2: Un autómata de estados finitos para la palabra "baa".

# Resumen teórico de capítulo 3

# Resumen teórico de capítulo 4

# Resumen teórico de capítulo 5