Class Notes Documentation

Release 0.1

Mario Rodas

May 04, 2012

CONTENTS

Disclaimer

Estos son mis apuntes personales de las diferentes clases en las que estoy inscrito. Solamente he tomado apuntes de las unidades que me parecian mas interesantes y/o considero merecían mayor profundidad. Aunque son unos apuntes en español, se nombraran los principales términos en inglés.

Cualquier sugerencia: `rodasmario2 <at> gmail <dot> com`

Contents:

CHAPTER

ONE

INTRODUCTION TO ARTIFICIAL INTELLIGENCE

Apuntes *incompletos* de ai-class.

1.1 Welcome to AI

Course Overview

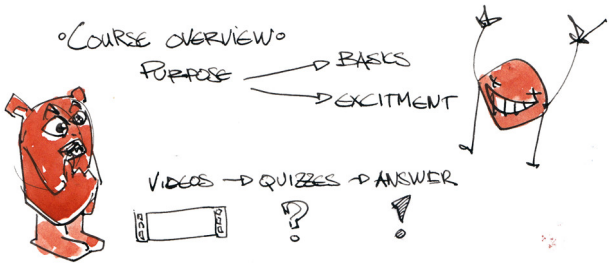

Figure 1.1: Course Overview (by larvecode)

Objetivos

- Enseñar las bases de Inteligencia Artificial
- Estimular el interes en Inteligencia Artificial

Estructura

1. Videos
2. Quizzes
3. Answer videos
4. Homework Assignment
5. Exams

1.1.1 Intelligent Agents

El agente puede percibir el entorno (*environment*) a traves de sus sensores (*sensors*) y puede modificar su estado mediate sus actuadores (*actuators*).

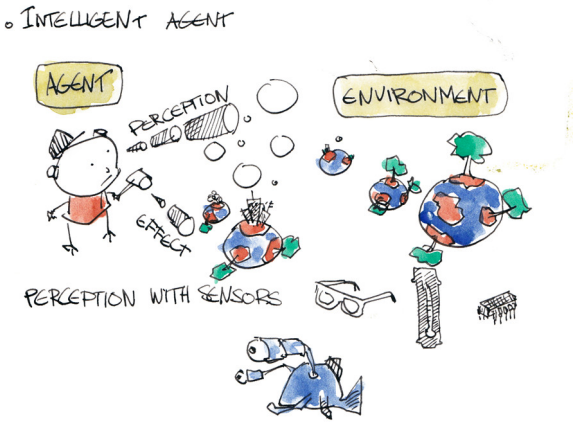

Figure 1.2: Intelligent Agent (by larvecode)

La principal pregunta en inteligencia artificial es la función que asigna los sensores a los actuadores. Esto es llamado politica de control (*control policy*)

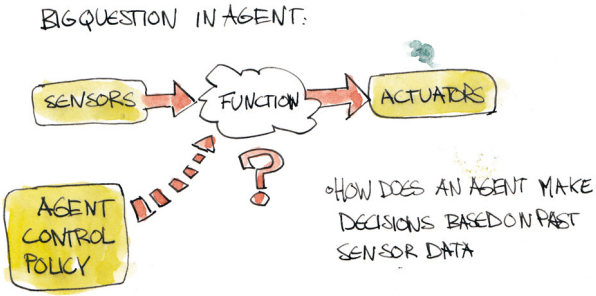

Figure 1.3: Agent Control Policy (by larvecode)

A lo largo de esta clase aprenderemos cómo el agente puede hacer mejores decisiones para transportar a sus actuadores, basandose en los datos del sensor.

Las decisiones llevan a cabo muchas, muchas veces, en un bucle del *feedback* del entorno, decisiones de los agentes, interaccion de los actuadores con el entorno y asi. Esto es llamado el ciclo de percepción-accion (*perception action cycle*).

1.1.2 Aplicaciones de Inteligencia Artificial

AI in finance

Existe un gran número de aplicaciones de inteligencia artificial en finanzas, a menudo en la forma de toma de decisiones comerciales –en cuyo caso el agente se llama agente de comercio(*trading agent*)–. Y el entorno podrían ser el mercado de valores, el mercado de bonos o el mercado de materias primas, también puede leer noticias en linea y seguir ciertos eventos. Y sus decisiones son usualmente decisiones de compra o venta.

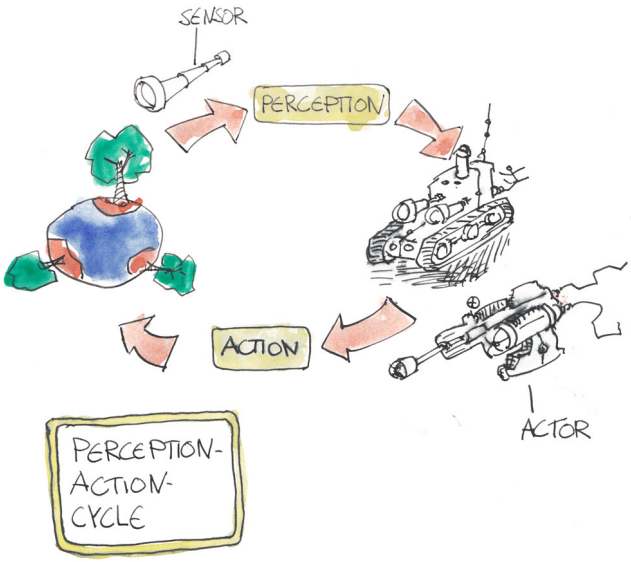

Figure 1.4: Perception Action Cycle (by larvecode)

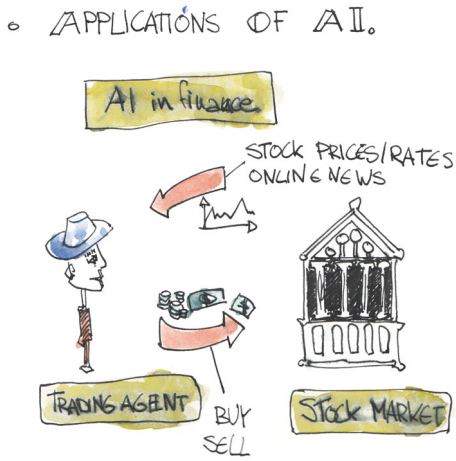

Figure 1.5: AI in finance (by larvecode)

AI in robotics

Hay una gran historia de inteligencia artificial con la robotica. hay diferentes tipos de robot y como ellos se relacionan con el entorno mediante sus sensores (camaras, micrófonos, sensores tactiles); y como ellos reaccionan a su entorno (mover sus ruedas, brasos, etc.)

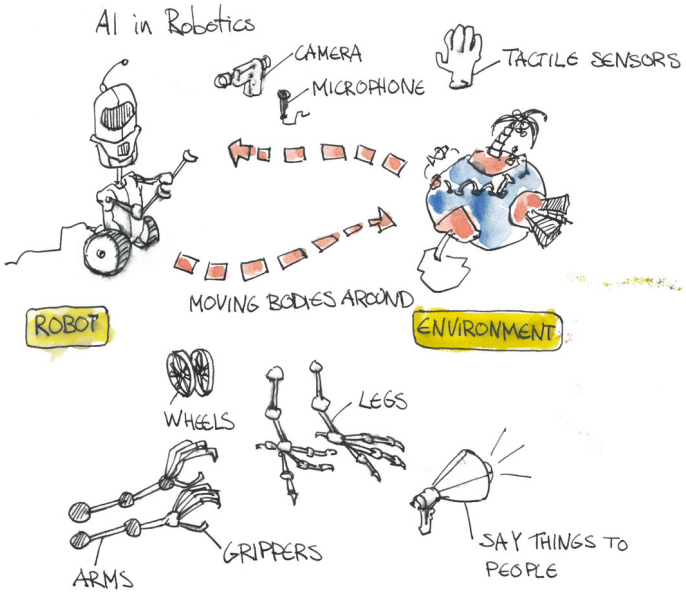

Figure 1.6: AI in robotics (by larvecode)

Web crawlers

Cuando surgió internet, los primeros *web crawlers* se llamaron *robots*, y para bloquearles el acceso a tu página web, hay un archivo `robots.txt`.

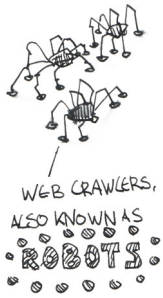

Figure 1.7: web crawlers (by larvecode)

AI in games

Hay dos maneras en la que los juegos usan inteligencia artificial:

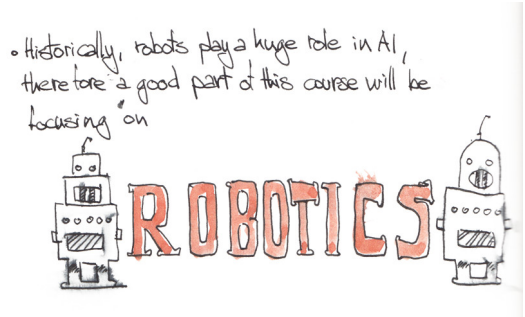

Figure 1.8: AI in robotics (by larvecode)

Play against you

Una de las maneras es cuando el *game agent* juega contra ti, como si fuera un ser humano.

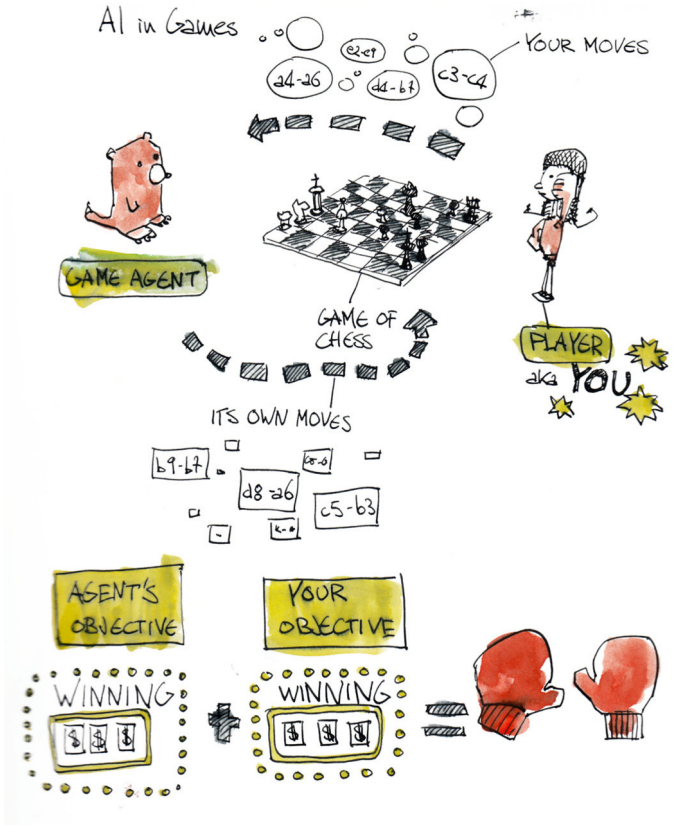

Figure 1.9: AI in games (by larvecode)

Make characters in game more believable

Figure 1.10: AI in games (by larvecode)

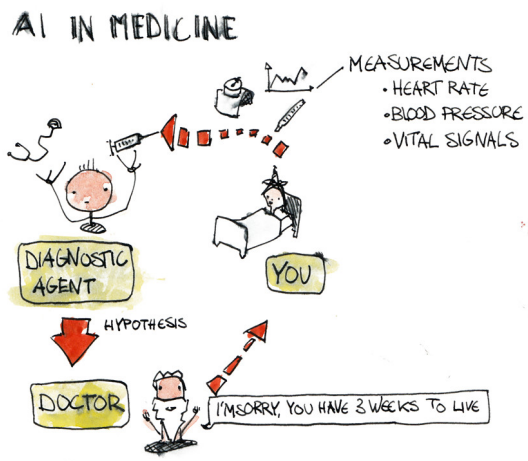

Figure 1.11: AI in medicine (by larvecode)

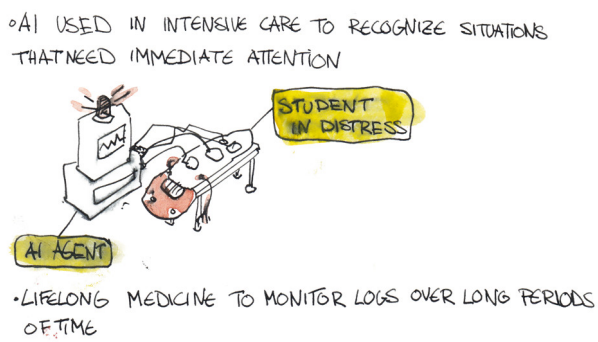

Figure 1.12: AI in medicine (by larvecode)

AI in medicine

AI on the web

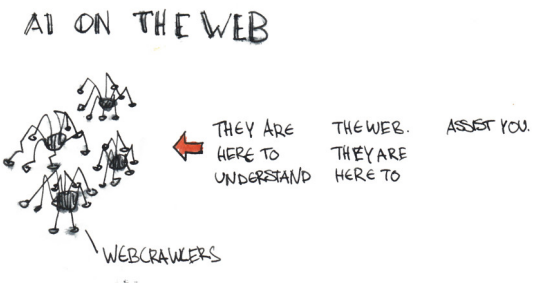

Figure 1.13: AI on the web (by larvecode)

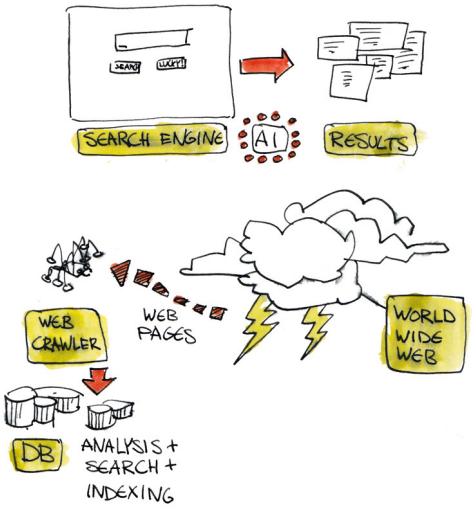

Figure 1.14: AI on the web (by larvecode)

1.1.3 Terminología

Existen diferentes terminos que se usan para distinguir los problemas de Inteligencia Artificial.

Fully vs. Partially Observable

Deterministic vs. Stochastic

Discrete vs. Continuous

Benign vs. Adversorial

environment

fully observable (completamente observable) Cuando lo que el agente puede detectar en cualquier punto del tiempo es información suficiente para tomar una decision optima.

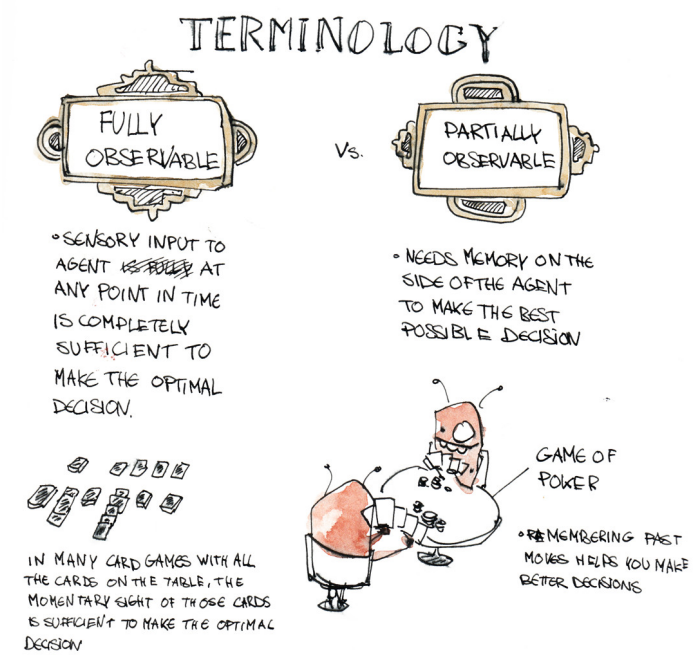

Figure 1.15: fully vs. partially observable (by larvecode)

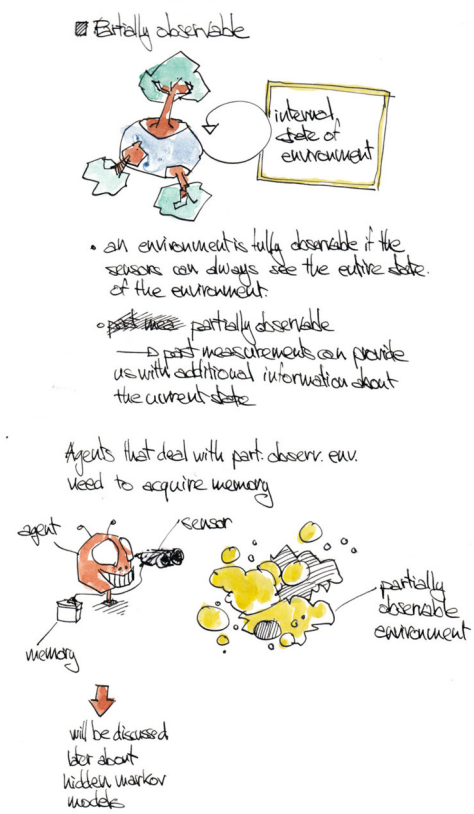

Figure 1.16: partially observable (by larvecode)

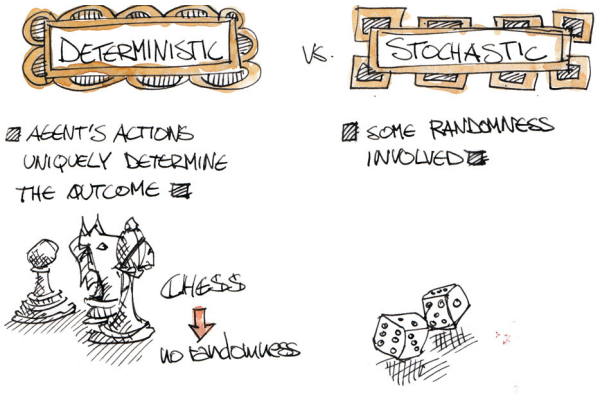

Figure 1.17: deterministic vs. stochastic (by larvecode)

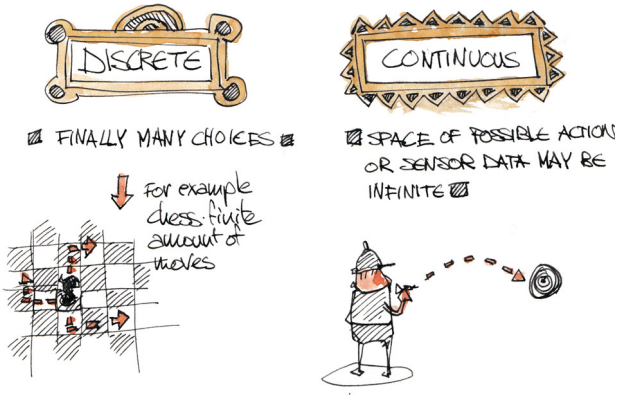

Figure 1.18: discrete vs. continuous (by larvecode)

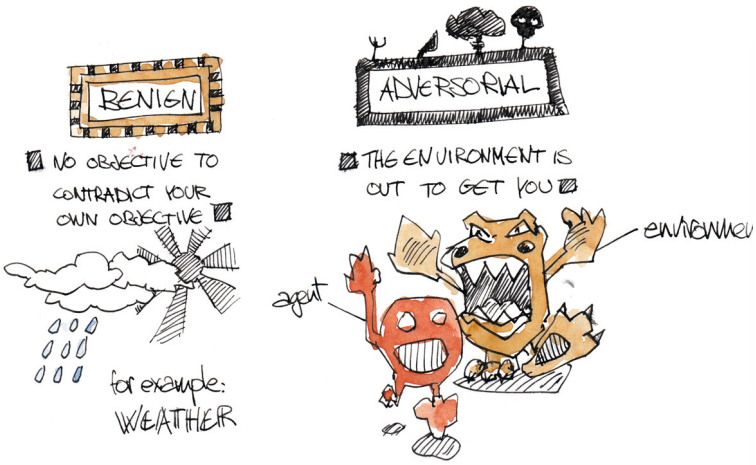

Figure 1.19: benign vs. adversorial (by larvecode)

partially observable (parcialmente observable) Cuando el agente es conciente solamente de una parte de estado del *environment*

deterministic (determinista) Cuando las acciones de tu agente unicamente determina el resultado.

stochastic (estocastico) cuando

discrete Cuando

continuous Cuando

1.1.4 AI and Uncertainty

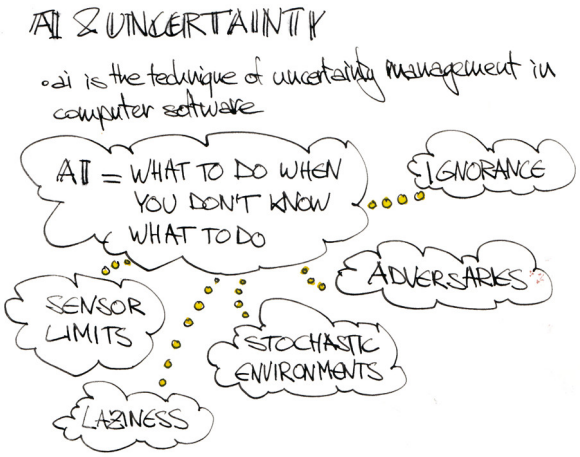

Figure 1.20: AI & Uncertainty (by larvecode)

1.1.5 Machine Translation

1.1.6 Summary

1.2 Problem Solving

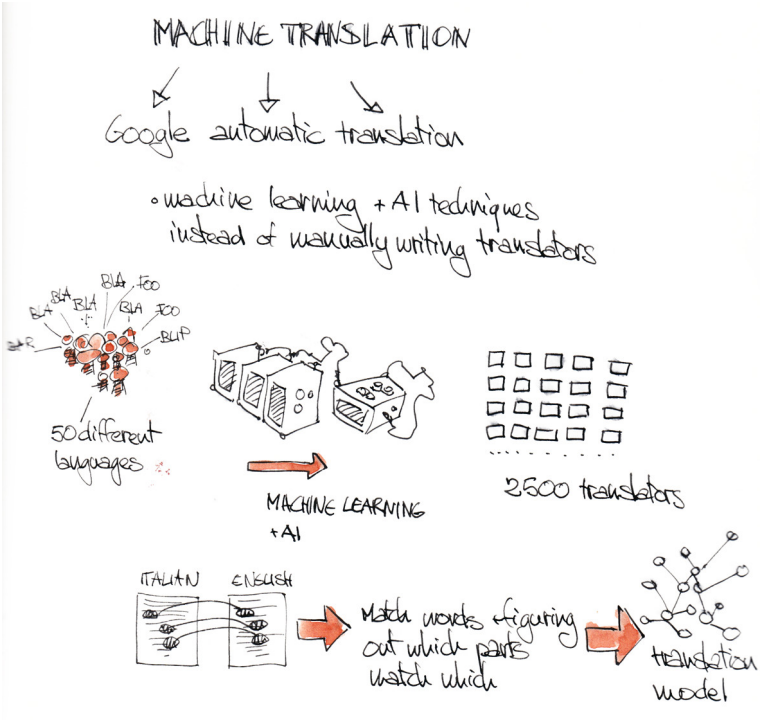

Figure 1.21: Machine Translation (by larvecode)

CHAPTER

TWO

MACHINE LEARNING

Apuntes *incompletos* de ml-class.

2.1 Introduccion a Machine Learning

2.1.1 Definicion

Machine Learning is ...

> ... a field of study that gives computers the ability to learn whitout bening explicitly programmed
>
> —Arthur samuel (1959)

> ... well-posed Learning Problem: A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.
>
> —Tom Mitchell (1998)

2.1.2 Unsupervised learning

> ... Aqui esta el dataset ¿puedes encontrar alguna estructura el los datos?

2.2 Linear regression with one variable

2.2.1 Model representation

2.2.2 Cost function

2.2.3 Cost funtion intuition I

2.2.4 Cost funtion intuition II

2.2.5 Gradient Descent

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

CHAPTER

THREE

INTRODUCTION TO DATABASES

Apuntes *incompletos* de db-class.

3.1 Introducción

Course Overview

Este curso trata acerca de base de datos y el uso de sistemas de gestion de base de datos (*Database Management System* [1]), desde el punto de vista del diseñador, usuario y desarrolador de aplicaciones de base de datos.

3.1.1 Database Management System

Provides...

... efficient, reliable, convenient, and safe multi-user storage of and access to massive amounts of persistent data.

massive los sistemas de bases de datos manipulan datos a escala masiva (*massive scale*). Si pensamos las cantidades de datos producidas hoy en día, los DBMS manipulan, hasta, terabytes [2] de datos cada día.

Un aspecto critcico de los datos que manipulan los DBMS, es más mucho más grande que el que pued ingresarse en una memoria de un sistema de computo típico. Así la capacidad de las mmorias estan creciendo muy, muy rápido, pero los datos en el mundo y los datos manipulados por los DBMS crecen mucho mas rápido.

Los DBMS están diseñados para manipular data que reside fuera de la memoria

Persistent Los datos manipulados por los DBMS es tipicamente persistente [3], lo que quiere decir que los datos *sobreviven* a los programas que se ejecutan en los datos. Por ejemplo, si ejecutas un programa de computadora, iniciara las variables que creamos, los datos estarán mientras el programa se ejecuta, y cuando el programa finalice los datos se *iran*. pero en las bases de datos los datos estan ahi, cuando el programa inicia y ejecuta sobre los datos y se detiene los datos seguiran ahí. Frecuentemente, muchos programas estarán operando sobre los mismos datos.

Safe Los DBMS ejecutan aplicaciones criticas tales como telecomuniccaciones y sistemas bancarios tienen que garantizar que los datos en la base de datos permanezcan en un estado consistente, no se perderán o sobreescribira conando ocurran fallas, y pueden haver fallas de hardware, software, cortes de energía, o usuarios maliciosos intentan corromper los datos.

[1] http://en.wikipedia.org/wiki/Database_management_system
[2] 10^{12} bytes. http://en.wikipedia.org/wiki/Terabyte
[3] http://en.wikipedia.org/wiki/Persistence_%28computer_science%29

Asi los DBMS poseen un numero de mecanismos incluidos que aseguran que los datos permanezcan consistentes, sin importar lo que suceda.

Multi-User Una aplicacio de usuario accede a los datos **concurrentemente**. Asi cuando tienes multiples aplicaciones trabajando sobre los mismos datosm el DBMS, tiene que poseer los mecanismos para asegurarse que los datos permanezcan consistentes. Por ejemplo, para que no tengamos la mitad de un item manipulado por un usuario y la otra mitad sobreescrita por otro usuario,

Asi, los DBMS, tienen un sistema llamado control de concurrencia (*concurrency control*)m y la idea es que nosotros controlamos la manera como multiples usuarios acceden a la base de datos. Nosotros no controlamos con un solo usuario que tiene acceso exclusivo a ala base de datos o la perfomance descendera considerablemente, asi el control ocurre, actualmente, al nivel de items de datos en la base de datos. Muchos usuarios pueden manipular la misma base de datos pero estaran operando en diferentes items de datos individuales.

Convenient Es una de las carateristicas criticas de la base de datos. Los DBMS estan diseñados para que sea fácial trabajar con grandes cantidades de datos y hacer pontentes y interesantes procesamient sobre los datos. Niveles:

Physical Data Independence la manera en que los datos estan almacenados y dispuestos en los discos es independiente a la manera en que los programas *piensan en* la estructura del dato.

High-Level Query Languages

- Compacto
- **Declarativo**: En el *query* describes lo que quieres de la base de datos, sin algoritmos.

efficient

JOKE *En bases de datos, las tres cosas más importantes son la perfomance, en segundo la perfomance y por último perfomance*

las bases de datos...

- Miles de consultas (*querys*) por segundo

reliable

- 99,9999 % uptime

3.1.2 Conceptos Claves

Data model Es una descripcion, en general, como los datos estan estructurados.

- Set of records
- XML -> jerarquico
- graph -> nodos y vertices.

Schema versus data

Schema

- tipos (*types*) en lenguajes de programación
- la estructura de la base de datos
- se define al inicio y no cambia
- DDL

Data

- variables en lenguajes de programación
- datos almacenados dentro del schema

- cambia rapidamente

Data Definition Language (DDL)

- Configura el schema/estructura de la bse de datos.

Data Manipulation Language (DML)

- Consultar y modificar la base de datos.

3.1.3 Personas claves

personas involucradas en un sistemas de base de datos

BDMS Implementer persona que implementa el DBMS -> contruye el sistema [fuera de los alcances de este curso]

Database designer persona que establece el schema de la base de datos

Database Application Developer persona que construye una aplicacion o programa que se ejecutara sobre la base de datos.

Database Administrator persona que carga (*load*) el DBMS y se encarga de que permanezca ejecutandose.

- trabajo interesante
- se encarga del perfomace
- bien pagado

Important: Lo sepas o no, usamos bases de datos cada dia. mejor dicho, cada hora.

3.2 The Relational Model

- Tiene ~35 años
- fundamento de las bases de datos, y subyace en la mayoria de bases dedatos comerciales
- Es un modelo muy sencillo y extremadamente expresivo para realizar *preguntas* a ala base de datos.
- Existe eficientes implementaciones del modelo relacional.

3.2.1 Construciones básicas del modelo relacional

Important: Database = set of named relations (or **tables**)

---

relation Una base de datos esta contituido por un conjunto de relaciones (*relations*) referidos como "tablas" (*tables*), los cuales tienen un nombre

Por ejemplo la tabla `Student` (en singular):

```
Student
+------+-------+--------+--------+
|      |       |        |        |
+======+=======+========+========+
|      |       |        |        |
+------+-------+--------+--------+
```

```
|       |        |         |         |
+------+-------+--------+--------+
         ...
+------+-------+--------+--------+
|       |        |         |         |
+------+-------+--------+--------+
```

Important: Each relation has a set of named attributes (or **columns**)

attributes Ejemplo en la tabla `Student`, agregamos los atributos: *ID*, *name*, *GPA*, *Photo*:

```
Student
+------+-------+--------+--------+
| ID   | name  | GPA    | Photo  |
+======+=======+========+========+
|       |        |         |         |
+------+-------+--------+--------+
|       |        |         |         |
+------+-------+--------+--------+
         ...
+------+-------+--------+--------+
|       |        |         |         |
+------+-------+--------+--------+
```

Important: Each tuple (or **row**) has a value for each attribute.

tuple Por ejemplo en nuestra tabla `Students`:

```
Student
+------+-------+--------+--------+
| ID   | name  | GPA    | Photo  |
+======+=======+========+========+
| 123  | Amy   | 3.9    | (^_^)  |
+------+-------+--------+--------+
| 234  | Bob   | 3.4    | (T_T)  |
+------+-------+--------+--------+
         ...
+------+-------+--------+--------+
|       |        |         |         |
+------+-------+--------+--------+
```

Important: Each attribute has a type (**domain**)

type Por ejemplo en nuestra table `Student`:

```
ID     --> integer
name   --> string
GPA    --> float
photo  --> jpeg file
```

3.3 Querying Relational databases

3.3.1 Relational Algebra

- **Formal**
- Simbolos griegos

ID of stundent with GPA > 3.7 applying to Stanford:

$$\pi_{ID}\sigma_{GPA>3.7\wedge cName=\text{'Stanford'}}(Student \bowtie Apply)$$

3.3.2 SQL

Actual/Implemented

ID of stundent with GPA > 3.7 applying to Stanford:

```
SELECT Student.ID
FROM Student, Apply
WHERE Student.ID=Apply.ID
AND GPA>3.7 AND cName='Stanford'
```

Important: Se recomienda aprender primero algebra relacional antes que SQL.

3.4 Well-formated XML

3.4.1 Contrucciones básicas

Tagged elements (nested) Por ejemplo, a continuación el tag es `first_name`:

```
<first_name>Mario</first_name>
```

Attributes Por ejemplo, a continuación el atributo es `ISBN`:

```
<book ISBN="ISBN-0-13-666">
   <!-- some content -->
</book>
```

Text Por ejemplo, a continuación el text el `Database systems`

```
<title>Database systems<title>
```

3.4.2 Modelo Relacional versus XML

.	Relacional	XML
Structure	Tables	Hierarchical, **Tree**, Graph
Schema	Fixed in advance	Flexible, "self-describing"
Queries	SQL, nice langs	Less so
Ordering	Unordered	Implied
Implementation	Native	Add-on

- En el modelo relacional el *Schema* es absolumanete requerido, en XML es opcional

3.4.3 Well-formed

requerimientos básicos

- single root element
- matched tags, proper nesting
- Unique attributes within elements

3.4.4 eXtensible Markup Language (XML)

- ... un estandar para la representacion y intecambio de datos
- ... competidor al modelo relacional
- ... más flexible que el modelo relacional
- ... la escepficicación formal de XML es enorme

3.5 DTD, ID & IDREFs

3.5.1 "Valid XML"

- Well-formed XML
- content-specific specification

Document type Descriptor (DTD)

XML Schema Descriptor (XSD)

Es un lenguaje para especificarlos tipos de los elementos, atributos, cuales permiten anidacion (*nested*), ocurrencias. También atributaos especiales como el ID y el IDREFs, que son una especie de *pointers*

Table 3.1: DTD/XSD versus "Well-formed" XML

DTD/XSD	well-formed XML
Programas asumen una estructura	flexibilidad
facilitan el CSS/XSL	Editar DTD can be painful
Permite el intercambio (especificación)	Leer DTD can be painful
Documentación	
Beneficios del "typing"	

3.5.2 Validando XML

Para validar DTD, podemos usar xmllint:

```
$ xmllint --valid --noout example.xml
```

CHAPTER

FOUR

NATURAL LANGUAGE PROCESSING

Apuntes *incompletos* de **nlp-class**_.

4.1 Basic Text Processing

4.1.1 Regular Expressions

> Regular expressions consist of constants and operators that denote sets of strings and operations over these sets, respectively.
>
> —From Wikipedia[#regexwiki]_:

Regular expressions are a formal language for specifying text string. In general, regular expressions provides a flexible mean to *match* strings of text. Commonly abbeviated as **regex** and **regexp**.

Metacharacter	Description
`.`	Match any character.
`+`	Match the preceding pattern element **one o more times**.
`?`	Match the preceding pattern element **zero o one times**.
`*`	Match the preceding pattern element **zero o more times**.
`{M,N}`	Denotes the minimun *M* andthe maximun *N* match count.
`[...]`	Denotes a set of possible character matches.
`|`	Separates alternate possibilities.
`^`	Initial of line.
`$`	Final of line.

regex	matches
[Aa]	amor, Amor
[123456790]	Any digit, or simply *[0-9]*

Books

Regular Expression Pocket Reference, 2nd Edition http://shop.oreilly.com/product/9780596514273.do

4.1.2 Word Tokenization

Task in NLP needs to do text normalization:

1. Segmentatio/tokenizing words in running text.

2. Normalizing word formats.
3. Segmenting senttences in running texts.

Concepts

Type An element of the vocabulary. Represented by N

Token An instance of that type running text. Represented by V. The size of the vocabulary is represented by $|V|$

Corpora Data sets of text.

- Generally in an sentence #tokens >= #types
- Chuch and Gale (1990): $|V| >= O(N^{1/2})$

Tokenizing, *first steps*

Using Unix Tools :-), we use *big.txt*, is not exactly the same of the class[#file]_:

```
$ curl -O http://norvig.com/big.txt
```

Replacing all non alphabetic characters with a newline (*n*), and display only the first 10 lines (*head*):

```
$ tr -sc 'A-Za-z' '\n' < big.txt | head
```

Sort the output:

```
$ tr -sc 'A-Za-z' '\n' < big.txt | sort | head
```

Merging upper and lower case:

```
$ tr 'A-Z' 'a-z' < big.txt | tr -sc 'A-Za-z' '\n' | sort | uniq -c
```

Sorting the counts:

```
$ tr 'A-Z' 'a-z' < big.txt | tr -sc 'A-Za-z' '\n' | sort | uniq -c | sort -n -r
```

Issues in tokenization

Apostrophe

Finland's capital -> Finland, Finlands, Finlands'
I'm -> I am

language issues

French

L'ensemble -> one token or two?
L?, *L'*?, *Le*?
Want *l'ensemble* to match with *un ensemble*.

4.1.3 Word Normalization and Stemming

Normalization

4.1.4 Sentence Segmentation

4.2 Edit Distance

4.2.1 Edit Distance

Edit distance is ...

> ... given two character strings s_1

4.2.2 How to compute edit distance?

Dynamic Programming A tabular omputation $D(n, m)$

Compute $D(i, j)$ for all i ($0 < i < n$) and j ($0 < i < m$).

Minimun Edit Distance (levinshtein)

CHAPTER

FIVE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*