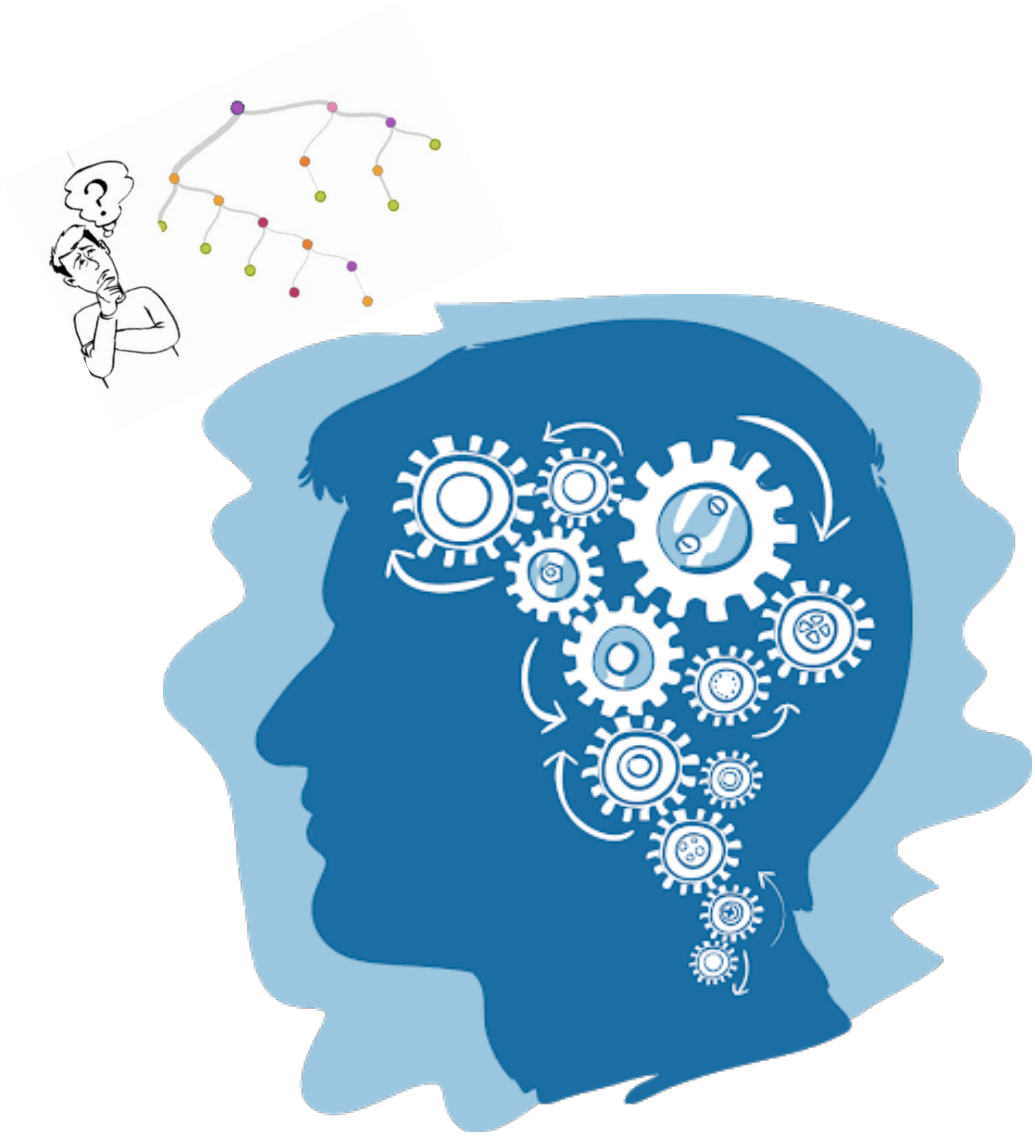
11 DE MAYO DE 2021



IMPLEMENTACIÓN DEL ALGORITMO ID3

PRÁCTICA 2 – INGENIERÍA DEL CONOCIMIENTO

VITALIY SAVCHENKO
CARLOS LUENGO HERAS
VICTOR VELASCO ARJONA

ÍNDICE

1. Algoritmo: En esta primera parte de la memoria trataremos de exponer la lógica detrás de nuestro algoritmo, como lo hemos implementado y que hemos utilizado para para ello.

Nuestro algoritmo parte del concepto de **nodo**, actúa sobre un conjunto de nodos al que llamaremos **árbol**, que realmente es un Set<node> que tratamos como un árbol.

Un nodo guarda información como su nombre (la conjugación del atributo) y un Set con sus hijos. Así, cada nodo formará parte del árbol de decisión final.

```
public class node {
    private String nombre;
    private Set<node> nodos;
```

Una vez entendido que como es nuestro árbol, podemos hablar de la lógica que hemos implementado para hacer funcionar nuestra versión del algoritmo ID3:

Cabe destacar de nuestra implementación dos estructuras claves: una lista (méritos, ganancias) donde guardamos el merito y la ganancia de elegir cada atributo para optimizar la decisión del árbol final; y una segunda lista (queue) donde vamos evaluando al siguiente nodo a incluir en el árbol.

Además, tenemos otros campos con menos importancia, como columnas, para mantener la cuenta de los atributos a evaluar, y un array de booleanos done para saber si ya hemos incluido ese nodo en el árbol.

```
public class ID3 {
    private Tabla tabla;
    private List<Pair> meritos, ganancia;
    private Boolean[] done;
    private node arbol;
    private Queue<node> queue;
    private int columnas;
```

Finalmente, una tabla que realmente es una matriz con las lecturas de los .txt organizadas para trabajar sobre ellas.

```
public class Tabla {
    private String[][] tabla;
    private Map<String, Double> veces, positivos;// numero de veces que se repite un string
    private List<HashSet<String>> tipos;// numero de diferentes string que hay en una columna
    private String[] nombres;
    private int cont;
```

Una vez entendida la estructura lógica el algoritmo es muy sencillo, tras leer los archivos de entrada rellenamos una matriz con todos los datos leídos, además de distintas estructuras que nos ayudarán a mantener las cuentas para cálculos de entropía, ganancia y mérito. Esto desencadena una serie de llamadas (patrón modelo--controlador) que acaba ejecutando el algoritmo, añadiendo al árbol de decisión los nodos y atributos que más influyen en la decisión final que se plantea en este algoritmo. Finalmente, printeamos el árbol con una función que hemos creado. Para una fácil corrección, el ejecutable crea un salida.txt, en vez de tener que ejecutarlo y estudiar los resultados por la consola.

2. Metodología aplicada: En esta segunda parte de la memoria trataremos de describir los detalles propios de nuestra implementación, empezando por el lenguaje que hemos utilizado, continuando con el diseño del nuestro código y otros patrones utilizados para estructurar su funcionamiento, y finalmente revisando las ampliaciones realizadas.

a) Lenguaje utilizado:

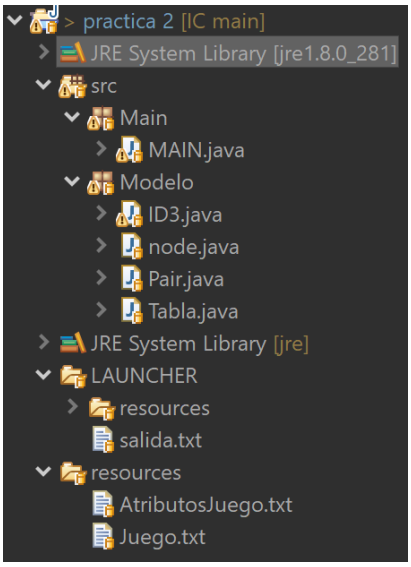
Para realizar esta práctica hemos decidido que la mejor opción sería utilizar Java. Nos ha parecido la mejor opción al ser una aplicación de escritorio, aunque estuvimos tanteando otras opciones como HTML-CSS y hacer una aplicación web ya que podíamos printear árboles de manera muy sencilla y visual.

Por lo tanto, el código se ha desarrollado utilizando un repositorio de GitHub y de editor la herramienta Eclipse. Además, aprovechando la versatilidad de Java y los paquetes de sus proyectos, hemos decidido utilizar el patrón modelo - controlador, aunque ya lo explicaremos en detalle en el punto siguiente.

b) Diseño del código:

Comenzamos con el mítico patrón modelo -controlador, de esta forma tenemos 2 paquetes principales:

La aplicación se estructurará así:



El controlador, que hemos llamado Main dado que no había vista, genera la primera instancia de la aplicación.

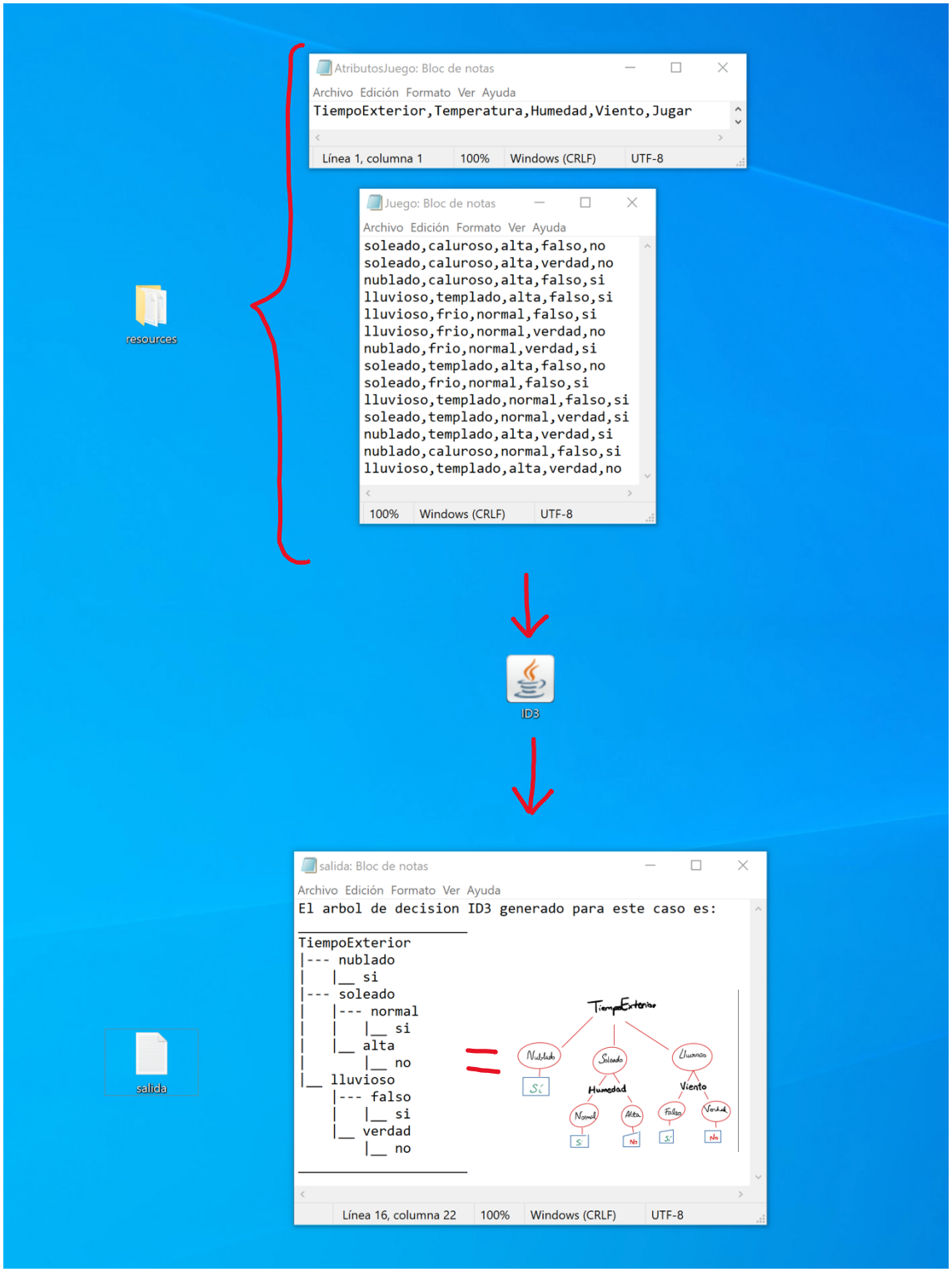
Modelo realizará la lectura de los archivos y procederá a ejecutar el algoritmo a través de los datos obtenidos.

Modelo generará un árbol de decisión óptimo que a su vez printeará gracias a una función que hemos hecho de recorrido del árbol con un iterador.

La carpeta resources cuenta los dos .txt reemplazables a gusto. Launcher cuenta con los resources, con un .jar (el que ejecutaremos en el manual), y con la salida que genera la ejecución de ese .jar.

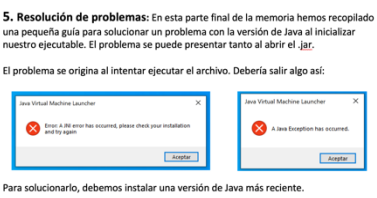
c) Ampliaciones realizadas:

Hemos realizado todas las ampliaciones. La aplicación implementa ***todos los niveles de recursividad.*** También hemos comprobado ***el correcto funcionamiento del algoritmo para los ejemplos de la tabla.***

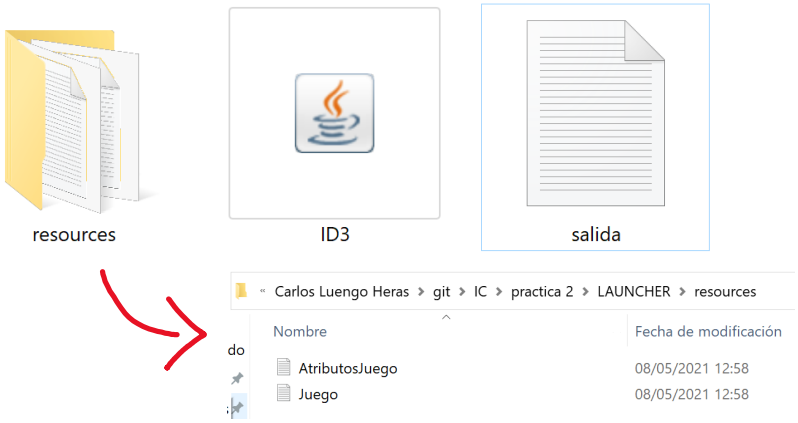


3. Manual de usuario: En esta parte de la memoria haremos un pequeño recorrido por la funcionalidad de la aplicación, a modo introducción a la aplicación. A diferencia de la práctica anterior simplemente es ejecutar el .jar entregado.

1- **Requisitos previos de Java:**
Verificar que la aplicación es ejecutable. Es de vital importancia que el sistema operativo tenga instalado una versión del jdk-jre actualizada y compatible. Si saltara algún error, por favor, revisa la resolución de problemas explicada al final de la memoria y en el manual.

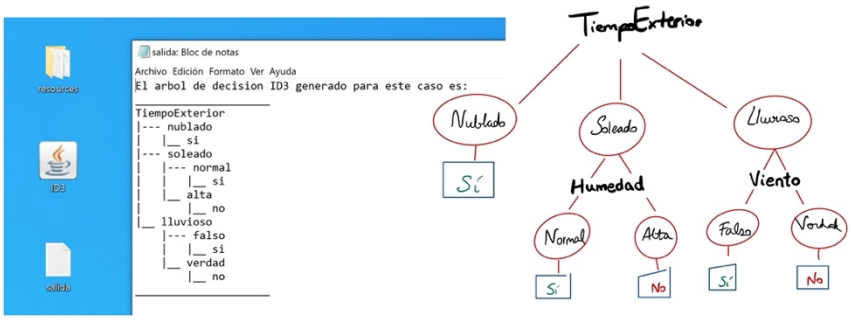


2- **Requisitos para el funcionamiento:**
El programa necesita localizar la carpeta resources con sus 2 archivos de texto, de los que extrapolar los datos y poder hacer el árbol de decisión. Si no tiene esta carpeta con sus dos archivos de texto, no funcionará correctamente.



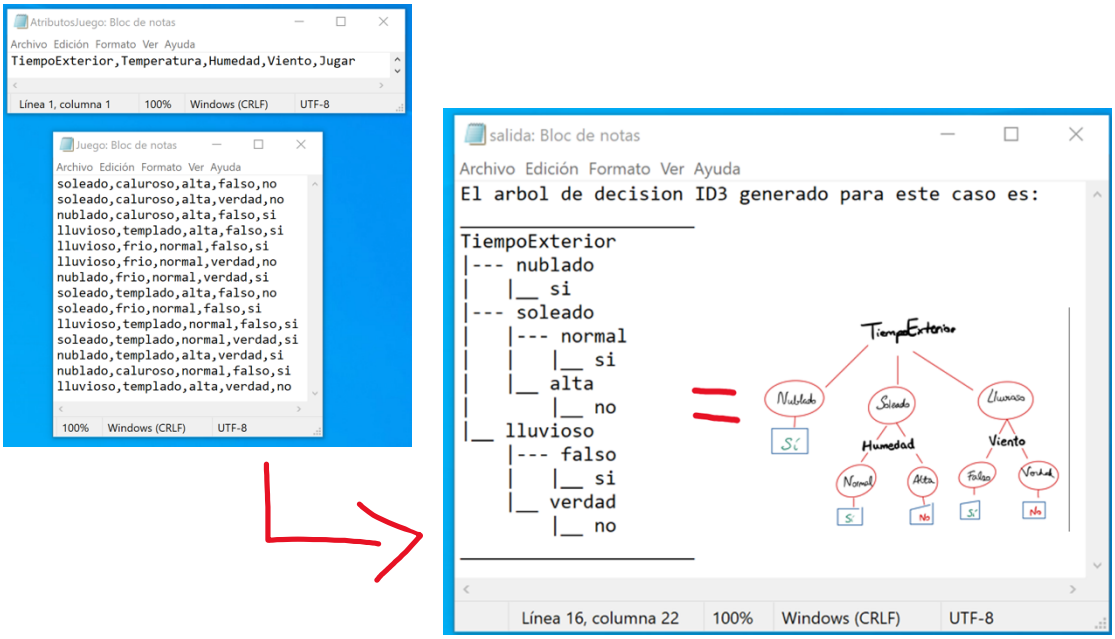
Importante: para que el algoritmo funcione correctamente, en caso de que algún atributo utilice las configuraciones "si" y "no", deberá cambiarse a "Si" y "No" para que no entre en conflicto con las variables "si" y "no" de resultado.

3- **Resultado:**
Finalmente, al ejecutar el ID3 se generará un salida.txt que contendrá el árbol. La representación en Java no existe, hemos ideado una forma de recorrerlo que lo muestra por consola (y lo imprime en un .txt), y a la derecha lo que significa.



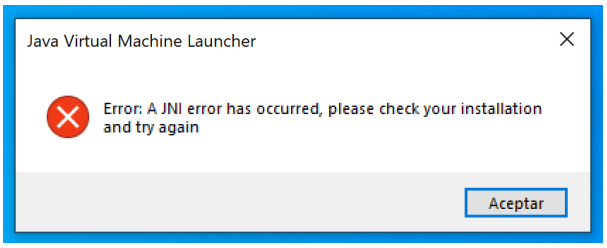
4. Ejemplos de compilación: En esta parte de la memoria hemos recopilado ejemplos de casos de uso para demostrar que la aplicación funciona correctamente sin necesidad de hacer pruebas recurrentes.

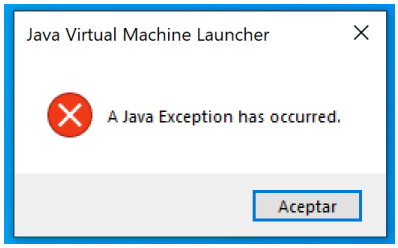
Un ejemplo básico muy conocido del algoritmo, donde evaluamos la posibilidad de ir a jugar al fútbol en función de distintas condiciones climatológicas. Entran en juego el Tiempo Exterior (soleado, nublado o lluvioso), con la Temperatura (caluroso, templado o frío), la Humedad (alta o baja) y el Viento (verdad o falso).



5. Resolución de problemas: En esta parte final de la memoria hemos recopilado una pequeña guía para solucionar un problema con la versión de Java al inicializar nuestro ejecutable. El problema se puede presentar tanto al abrir el .jar.

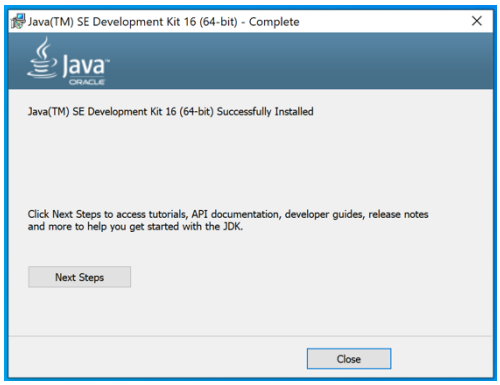
El problema se origina al intentar ejecutar el archivo. Debería salir algo así:





Para solucionarlo, debemos instalar una versión de Java más reciente.
En este caso, la podremos descargar desde el siguiente enlace de la web oficial de Oracle:

https://www.oracle.com/java/technologies/javase-jdk16-downloads.html



Una vez instalado, podemos ejecutar el .jar correctamente.