Enunciados de las entregas de la asignatura de IAC

Grado en Ingeniería de Sistemas de Información
Tercer curso
Profesor: Mariano Fernández López
Escuela Politécnica Superior, Universidad San Pablo CEU

Índice general

1. Prácticas **2**
1.1. Introducción 2
1.2. Práctica de lógica computacional 2
1.2.1. Pasos a realizar 2
1.2.2. Material a entregar 3
1.3. Práctica de búsqueda heurística sin adversarios 4
1.3.1. Pasos a realizar 4
1.3.2. Material a entregar 5
1.4. Práctica de aprendizaje automático 6
1.4.1. Pasos a realizar 6
1.4.2. Material a entregar 6
1.5. Criterios de evaluación 7
1.6. Forma de entrega 7

2. Licencia **8**
2.1. Licencia Apache versión 2 8

Capítulo 1

Prácticas

1.1. Introducción

En las siguientes secciones se muestran las prácticas a desarrollar durante el curso. Habrá una sobre lógica computacional (sección 1.2), otra sobre búsqueda heurística (sección 1.3) y otra sobre aprendizaje automático (sección 1.4).

Además de leer el enunciado de cada práctica particular, es importante que lea las secciones 1.5 y 1.6 así como el capítulo 2.

1.2. Práctica de lógica computacional

El propósito de esta práctica es que alumno desarrolle un prototipo de sistema en Prolog para resolver un problema que se pueda representar de forma natural mediante reglas.

1.2.1. Pasos a realizar

Son los que se muestran a continuación:

1. *Elegir el problema a resolver.* Puede ser la representación de las reglas de un juego sencillo (p.ej., las tres en raya), o un subconjunto de reglas de un juego complejo (p.ej. algunas reglas del juego del ajedrez), un asistente para problemas escolares (p.ej. un asistente para la formulación química), o cualquier otro problema del que el estudiante conozca las reglas o tenga una persona cercana y accesible que las conozca. Por ejemplo, si es posible disponer del conocimiento de un médico, también puede estar orientado el sistema a este dominio.

2. *Describir las reglas del sistema en lenguaje natural.* A modo de ejemplo, supóngase que se desea desarrollar un sistema para comprobar la

validez de dobles máster. Un conjunto de reglas adecuado al alcance previsto en esta práctica es el que se muestra a continuación:

> *a*) Cobertura del máster 1: para toda asignatura del máster 1, bien pertence el conjunto de asignaturas del doble máster, bien tiene una equivalencia con alguna asignatura del doble máster.
>
> *b*) Cobertura del máster 2 (análogo al paso anterior).
>
> *c*) Precisión del doble máster: para toda asignatura del doble máster, bien pertenece al máster 1 bien pertenece al máster 2.
>
> *d*) Secuenciación correcta: no se da el caso de que una asignatura aparezca en distinto semestre en un máster simple y en el máster doble.

3. *Elaborar un modelo conceptual para representar las entidades más relevantes cuya existencia se asume en las reglas anteriores* (opcional). Como ejemplo, se puede examinar el modelo conceptual de la figura 1.1.

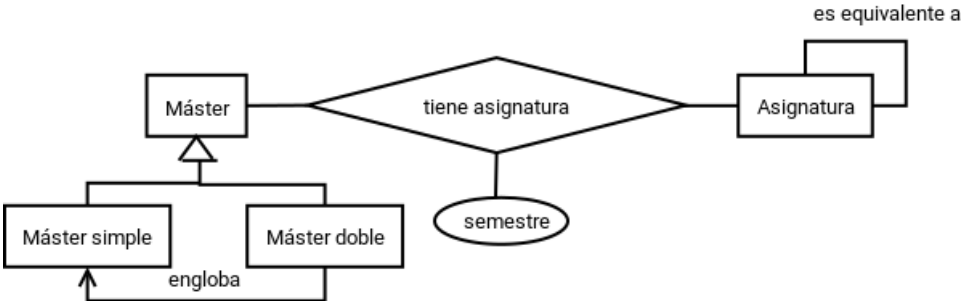

Figura 1.1: Modelo conceptual del problema del doble máster

4. *Representar en Prolog una situación concreta que se ajuste al modelo conceptual que ha descrito en el diagrama.* A modo de ejemplo, véanse los hechos que aparecen el fichero proporcionado por el profesor.
5. *Codifique en Prolog las reglas anteriores.* El ejemplo del código mencionado anteriormente contiene las reglas del del validador del doble máster.

1.2.2. Material a entregar

Deberá ser el siguiente:

1. Código en Prolog.
2. Fichero README.md en formato Markdown mostrando el resultado y la justificación de tal resultado de cada uno de los pasos del apartado anterior.
3. Fichero LICENSE con el contenido indicado en la sección 2.1 del presente enunciado.

1.3. Práctica de búsqueda heurística sin adversarios

El propósito de esta práctica es que el alumno reutilice código que implementa el algoritmo A*. Se bajará código de Github, y lo utilizará para ejecutar este algoritmo para un grafo concreto.

1.3.1. Pasos a realizar

En primer lugar, debe bajarse de Github el código de algoritmos y estructuras de datos en Java de Justin Wetherell:

```
git clone https://github.com/phishman3579/java-\
algorithms-implementation.git
```

A continuación, examine el código de la clase *AStar.java*, más concretamente, el método *aStar*. Luego, examine los tests del A* que hay en:

```
java-algorithms-implementation/test/com/jwetherell/algorithms/graph/test/
    Graphs.java
```

Después, de forma gradual, desarrolle la clase principal donde estará el programa de la práctica. Comience con un código sencillo:

```
package aplicacion;

public class Main
{
    public static void main(String[] args)
    {
        System.out.println("Esto es una prueba");
    }
}
```

Es muy importante que no olvide la línea

```
package aplicacion;
```

Para poderlo compilar y ejecutar sin necesidad de crear ningún proyecto en un IDE (p.ej. Netbeans), modifique el fichero *build.xml* de la siguiente forma[1]:

[1]Para entender el papel de los fichero *build.xml* en Java y el uso de *ant* se recomienda que consulte el manual de Apache accesible a través de la siguiente dirección: `https://ant.apache.org/manual/tutorial-HelloWorldWithAnt.html`

En la zona indicada por *<!– set global properties for this build –>*, se debe añadir la siguiente línea de código:

```
<property name="main-class"  value="aplicacion.Main"/>
```

El *target* de *dist* también se debe modificar para que quede de la siguiente manera:

```
<target name="dist" depends="build" description="generate the
    distribution">
    <jar jarfile="${dist}/java-algorithms-implementation-${DSTAMP}.jar"
        basedir="${build}">
        <manifest>
            <attribute name="Main-Class" value="${main-class}"/>
        </manifest>
    </jar>
</target>
```

Luego, se añade:

```
<target name="run_main" depends="dist">
     <java jar="${dist}/java-algorithms-implementation-${DSTAMP}.jar"
           fork="true"/>
</target>
```

A continuación se comprueba que el programa funciona:

```
ant run_main
```

Después, tomando como referencia los tests, se crea un programa principal que genere un camino aplicando A*.

A continuación, responda a las siguientes preguntas:

1. ¿Qué variable representa la lista ABIERTA?
2. ¿Qué variable representa la función g?
3. ¿Qué variable representa la función f?
4. ¿Qué método habría que modificar para que la heurística representara la distancia aérea entre vértices?
5. ¿Realiza este método reevaluación de nudos cuando se encuentra una nueva ruta a un determinado vértice? Justifique la respuesta.

1.3.2. Material a entregar

Deberá ser el siguiente:

1. El proyecto completo, es decir, lo que se ha bajado inicialmente de Github más el código implementado.
2. `README.md` en formato Markdown respondiendo a las preguntas que se plantean en el presente enunciado.

1.4. Práctica de aprendizaje automático

1.4.1. Pasos a realizar

Los pasos a realizar son los que se muestran a continuación:

1. *Elegir el problema.* La predicción de resultados de partidos en la ATP o en la WTA de tenis, una liga deportiva, una competición de deporte electrónico, o cualquier otro problema del que se dispongan de datos suficientes como para aplicar algoritmos de aprendizaje automático.

2. *Identificar la fuente de datos.* Es necesario disponer de una serie de datos históricos que sirvan para que el sistema aprenda.

3. *Identificar las características relevantes de los hechos.* Por ejemplo, en la predicción del resultado de un encuentro de tenis, algunas características importantes pueden ser la posición en el ranking de cada jugador, la superficie en que se juega, la edad de cada jugador, etc.

4. *Obtener un fichero .arff con los hechos codificados de acuerdo con las características anteriormente elegidas.* Este fichero servirá como entrada para la herramienta Weka[2].

5. *Evaluar distintos algoritmos de aprendizaje automático con los datos obtenidos.* Este paso se llevará a cabo con la herramienta Weka, y tendrá como salida el algoritmo con mejor rendimiento para los datos datos.

6. *Generar en Java un objeto persistente con el algoritmo obtenido en el paso 5.* También se realizará con Weka.

7. *Implementar un prototipo de aplicación que consulte el objeto persistente generado en el paso 6.* La aplicación cargará en memoria el objeto persistente, que tendrá como responsabilidad la resolución del problema propuesto (p.ej. el pronóstico de un resultado deportivo), e interactuará con el usuario a través de una interfaz (que puede ser de texto) (véase la figura 1.2).

1.4.2. Material a entregar

Deberá ser el siguiente:

1. Cógido fuente en Java.

2. Los ficheros adicionales necesarios para poder compilar dicho código.

3. El fichero *.arff*.

[2] `http://www.cs.waikato.ac.nz/ml/weka/`

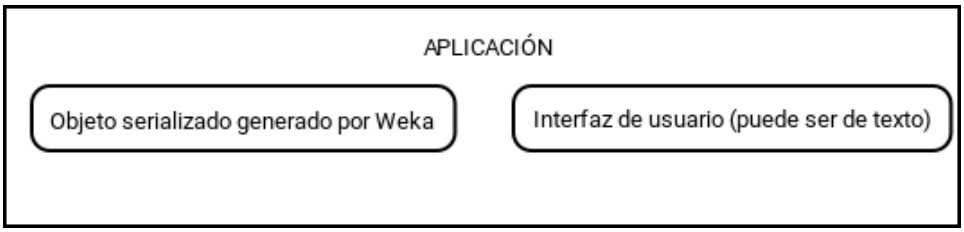

Figura 1.2: Estructura de la aplicación resultante de la práctica de aprendizaje automático

4. `README.md` en formato Markdown mostrando el resultado (o la referencia al resultado) y la justificación de tal resultado de cada uno de los pasos del apartado anterior. Deberá incluir también un ejemplo de ejecución que sirva de guía para quien desee utilizar el prototipo.

5. Fichero `LICENSE` con el contenido indicado en la sección 2.1 del presente enunciado.

1.5. Criterios de evaluación

Tanto el peso que tiene en la evaluación de la asignatura como el umbral de los distintos aspectos de la práctica están publicados en la guía docente.

A la hora de realizar la evaluación in situ, se aplicarán los siguientes criterios de corrección:

1. El producto debe funcionar correctamente para alcanzar la puntuación de aprobado.

2. Tendrá importancia la correcta aplicación de las técnicas de inteligencia artificial e ingeniería del conocimiento.

3. Se valorará tanto la documentación adecuada del código como el `README.md`.

4. Se valolará la usabilidad del producto.

1.6. Forma de entrega

Los materiales deberán ser subidos a un repositorio remoto (al que debe tener acceso el profesor)

Las fechas de las entregas serán publicadas por el profesor en el portal del alumno. **NO** se corregirá a partir del último *commit* dentro del plazo de entrega.

NO se permitirán formatos propietarios, por ejemplo, RAR.

Capítulo 2

Licencia

El estudiante será responsable de asegurar que las entregas se ajustan a la licencia explicada a continuación y que ha respetado las licencias de los recursos que haya reutilizado.

2.1. Licencia Apache versión 2

Los proyectos correspondientes a la práctica de representación del conocimientoy la de aprendizaje automático estarán bajo licencia Apache versión 2[1]. Se deberá incluir un fichero, llamado LICENSE con el siguiente contenido, disponible en la página Web de la licencia:

```
Copyright [año] [nombre del propietario del copyright]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

    Unless required by applicable law or agreed to in writing, software
    distributed under the License is distributed on an "AS IS" BASIS,
    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
    See the License for the specific language governing permissions and
    limitations under the License.
```

[1] https://www.apache.org/licenses/LICENSE-2.0