UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

Computación

Práctica Sistemas Multiagentes

Opción 1
Proceso ETL

José Carlos Gualo Cejudo

Enero, 2022

UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

(cont.)

Computación

Práctica Sistemas Multiagentes

Opción 1
Proceso ETL

Autor: José Carlos Gualo Cejudo

Tutor(a): Rubén Rodríguez Cardos

Enero, 2022

Notación y acrónimos

NOTACION

Ejemplo de lista con notación (o nomenclatura) empleada en la memoria del TFG.[1]

E	:	Proceso de Extracción
T	:	Proceso de Transformación
L	:	Proceso de Carga
.csv	:	Formato del archivo de salida
Python	:	Lenguaje de programación utilizado para la implementación
SPADE	:	Plataforma de sistemas multiagentes escrita en Python y basada en el envío instantaneo de mensanjes (XMPP)
XMPP	:	Estándar para el envío de mensajes y presencia

LISTA DE ACRÓNIMOS

Ejemplo de lista con los acrónimos empleados en el texto.

ETL	:	Extraction-Transform-Load
SPADE	:	Smart Python Agent Development Environment
XMPP	:	Extensible Messaging and Presence Protocol

[1] Se incluye únicamente con propósito de ilustración, ya que el documento no emplea la notación aquí mostrada.

Índice general

Índice de figuras

CAPÍTULO 1
Objetivo

Para la parte práctica de la asignatura de Sistemas Multiagentes, se pide el desarrollo completo de un sistema multiagente, desde su diseño hasta su implementación. De este modo, se ha llevado a cabo una solución imaginativa a través de el diseño y uso de este tipo de sistemas, que mediante la comunicación y la explotación adecuada de distintas fuentes de datos, logre la consecución del objetivo y las especificaciones que se explican a continuación.

1.1. ELABORACIÓN DE PROCESO ETL

Este proyecto consistirá en la creación de un sistema multiagente que simule un proceso ETL con el siguiente flujo:

(1) Extracción: Recolectar información de distintas fuentes de páginas de empleo. La información que se extraiga será la referente a: el sector, el numero de ofertas para ese sector en concreto, y la fuente utilizada. Este proceso se realizará mediante el uso de técnicas de Web Scraping para conseguir información procedente de las distintas páginas web de empleo. Originalmente se han utilizado seis fuentes de datos distintas, es decir, seis páginas web distintas cuyo código puede verse en el repositorio de la práctica, aunque finalmente solo se han utilizado cuatro de estas fuentes.

(2) Transformación: Transformar, filtrar, y estructurar la información obtenida. Para esta estructuración se han utilizado los tipos Dataframe. Y los datos debían filtrarse y transformarse según los siguientes criterios:
   - Se deben eliminar los sectores que tengan menos de 10 ofertas.
   - Se deben eliminar los sectores cuyo nombre tenga un nombre de más de 20.
   - Se debe eliminar los sectores que no estén en al menos dos fuentes de datos distintas.
   - Los textos no pueden contener:
     - Caracteres especiales, como acentos o signos de puntuación.
     - Espacios en blanco al empezar o al terminar.
     - En caso de tratarse de un número estos no pueden aparecer con 0's de relleno, por ejemplo: 08 se tendría que transformar en 8.

(3) Carga: Cargar la información obtenida del proceso anterior en un almacenamiento; en este caso se ha elegido un archivo con la extensión .csv.

A continuación se concretaran los aspectos fundamentales que han sido utilizados para el desarrollo del sistema multiagente que cumpla con los objetivos que hemos descrito, y con las necesidades que se han especificado.

CAPÍTULO 2

Arquitectura del Sistema

En este capítulo se detalla la arquitectura de los distintos agentes que se han empleado para desarrollar la práctica, así como explicar la estructura y comportamiento de cada uno de ellos.

Para ello, hablaremos de la arquitectura de cada uno de los agentes por separado, para finalmente ver la arquitectura final del sistema como un todo.

2.1. AQUITECTURA DE LOS AGENTES

2.1.1. Agente Extractor

Como se ha explicado en los objetivos de la práctica, el agente Extractor es el encargado de obtener la información en crudo de las distintas fuentes, (páginas web de empleo), para a continuación, enviar esa información a otro agente en el que se trate dicha información.

Bien, entrando más en detalle acerca de este agente, podemos decir que éste tiene un solo tipo de comportamiento, del cual van a existir 4 instancias, una por cada fuente de datos que se ha utilizado para la realización de la práctica. También es importante mencionar las diferentes funciones de las que va a disponer este agente, ya que cada una de estas funciones van a ser las encargadas de extraer la información de las distintas páginas de trabajo en forma de diccionario formateado a cadena de caracteres.

Estas funciones van a consistir sencillamente en algoritmos de Web Scraping para, mediante el análisis del código *HTML*, obtener datos acerca de los sectores que ofrecen esas páginas web, y el número de ofertas relacionadas con cada uno de esos sectores.

De este modo, cada una de las cuatro instancias que se crean del comportamiento usarán las funciones mencionadas para extraer la información; una vez obtenida esa información, el mismo comportamiento realizará el envío de un mensaje de tipo *Inform* especificando como lenguaje la fuente de la que han sido obtenidos los datos.

El destinatario de este mensaje es el Agente Transformador, del que hablaremos en el siguiente apartado.

2.1.2. Agente Transformador

Una vez que los datos han sido extraidos y han sido enviados por el Agente Extractor, es hora de recibir esos datos, y tratarlos para que puedan ser estructurados. Esa es la misión principal que tiene este nuevo agente.

Para poder llevar a cabo esto, se han utilizado tres comportamientos que vamos a pasar a explicar:

1. El primer comportamiento será el encargado de recibir los datos enviados por parte del Agente Extractor. El funcionamiento de este comportamiento es similar al que vimos para enviar los datos. De esta forma tenemos en marcha cuatro instancias de este tipo de comportamiento con el objetivo de recibir los cuatro mensajes que habían sido enviados desde el otro agente y almacenar los datos para su posterior conversión a Dataframe.

2. El segundo comportamiento se inicia cuando se el sistema reconoce que ya se ha obtenido toda la información que había sido enviada. Este comportamiento es el que lleva a cabo la función principal del Agente Transformador, que es la de transformar, filtrar, y estructurar los datos. Para ello, una vez que la información se ha recibido, la junta toda en un Dataframe, el cual se va a pasar por funciones que devolverán el Dataframe cumpliendo las diferentes restricciones y especificaciones mencionadas en el apartado del objetivo de la práctica.
3. Por último, tenemos el comportamiento encargado de enviar la información ya filtrada y estructurada al Agente de Carga. Para ello, cuando se finalice el comportamiento anterior, nacerá éste, del cual sólo existirá una instancia, y finalizará cuando el mensaje con la información final haya sido enviado.

Una vez enviado el mensaje con los datos de las páginas de empleo estructurados, limpiados, filtrados y estrucutrados, se finalizará el agente, y se dará inicio a la última acción que debe realizar el sistema, que se desarrollará en el siguiente punto.

2.1.3. Agente de Carga

Este Agente va a ser el encargado de exportar la información en un formato en el que dicha información pueda ser utilizada de manera sencilla por cualquier usuario; en este caso, el formato elegido ha sido *CSV*.

Para ello, Este agente consta de un comportamiento que recibe el mensaje con la información que había sido enviada en el punto anterior por el Agente Transformador, y la almacenará en un campo propio del Agente para su posterior exportación.

Y finalmente tendríamos el último comportamiento, que será el encargado de simplemente exportar el Dataframe obtenido a un archivo *CSV* en el directorio de trabajo.

2.2. ARQUITECTURA GLOBAL DEL SISTEMA

Tras haber explicado y entendido como funciona la arquitectura interna de cada uno de los agentes que componen el sistema, podriamos concluir en que la estructura del sistema sería como la descrita en la siguiente imagen.

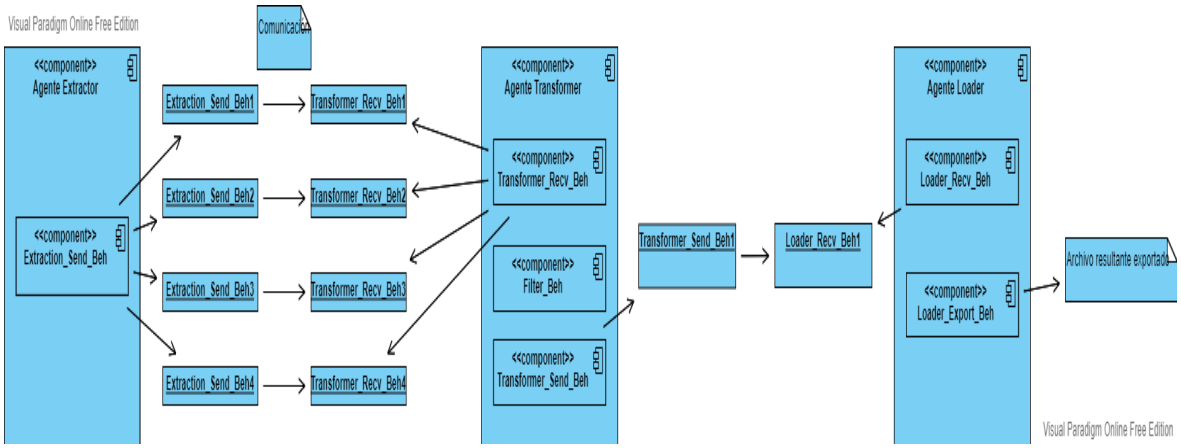

Figura 2.1: Arquitectura del sistema

CAPÍTULO 3

Toma de Decisiones

En esta sección se explicará brevemente el por qué de las diferentes decisiones que se han tomado en cuanto al diseño y arquitectura del sistema presentado en el capítulo 2, mostrando las razones que nos han llevado tomarlas, para acabar cumpliendo los objetivos mencionados en el cap. 1.

3.1. DECISIONES IMPORTANTES

En este apartado se describirán las diferentes decisiones que se han tomado, explicando porque se han realizado de esta forma.

- La primera decisión que se ha tomado, y que se debería tener muy en cuenta es la de tener diferentes instancias de comportamientos para el envío y la recepción de los mensajes. De esta manera, y como hemos explicado en el apartado anterior, los datos recogidos de cada fuente de información son enviados individualmente, siendo reconocido por su receptor a través de aplicar un filtro a los mensajes entrantes. El por qué de esta decisión se debe a que de este modo conseguimos:
  a) Mantener una adecuada organización en el intercambio de mensajes.
  b) Mantener un alto grado de modularización.
  c) No sobrepasar el peso máximo de los mensajes al utilizar diferentes vías.
- Otra decisión que se ha tenido que tomar es la de elegir el formato de salida que tendrá la información estructurada al acabar el proceso *ETL*. El formate que se ha elegido ha sido el *CSV*, ya que es un formato muy legible, y que puede ser analizado facilmente desde con su sencilla importación.
- Otra determinación que se ha realizado en el proyecto tiene que ver con el proceso de transformación. Como hemos dejado claro en el apartado de objetivos, una condición que tenían los resultados era que no hubiese caracteres extraños, como comas o acentos. Tras esto, a la hora de implementarlo, se plantearon dos opciones; ¿Eliminamos aquellos registros que tuvieran estos caracteres extraños, o los quitamos sin eliminar los registros? Bien, la decisión que se llevo a cabo fue la de mantener los registros eliminando simplemente los caracteres no deseados. La razón es que de esta manera se tendrían un mayor número de opciones para mostrar en el archivo resultante tras aplicar al conjunto de datos todos los filtros necesarios.

CAPÍTULO 4

Protocolos para la comunicación entre agentes

En este capitulo se aborda la descripción de los protocolos que se han utilizado para realizar la comunicación entre los agentes. Dicha comunicación se lleva a cabo mediante envío de mensajes en forma de cadena de texto. En nuestro caso, esas cadenas de texto van a consistir en diccionario que contengan la información de las páginas web, que posteriormente va a ser transformado a Dataframe.

En este sistema multiagente distinguimos principalmente dos procesos de comunicación que vamos a tratar de ver y explicar mediante diagramas de secuencia.

4.1. COMUNICACIÓN EXTRACTOR - TRANSFORMADOR

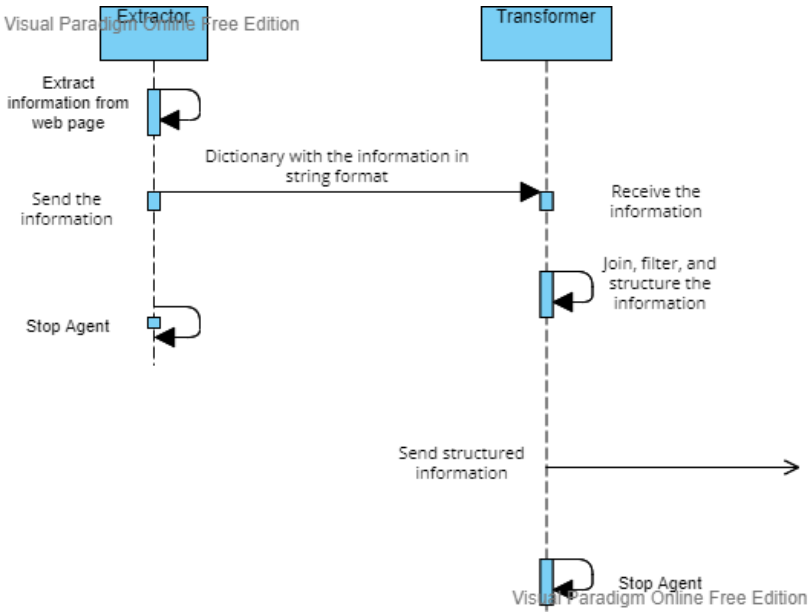

Figura 4.1: Diagrama de secuencia de la comunicación entre Agente Extractor y Transformador

Como vemos en la figura, el agente extractor extrae la información de las páginas web antes de ponerse en contacto con el agente Transformador. Es importante aclarar que este proceso se repite en los cuatro comportamientos que tiene el agente, que como se ha dicho anteriormente, cada uno es para una fuente de datos. Tras enviar los mensajes, éstos son recibidos por el agente transformador; y una vez que los ha recibido todos, los une para estructurarlos y aplicar los filtros que sean necesarios. Una vez que se han modificado, vuelven a ser enviados por el agente Transformador, como veremos a continuación. Por último, cabe mencionar la finalización del agente Extractor; ésta tiene lugar en el momento en el que todos sus comportamientos han acabado, es decir, cuando se ha enviado el último mensaje.

4.2. COMUNICACIÓN TRANSFORMADOR - CARGADOR

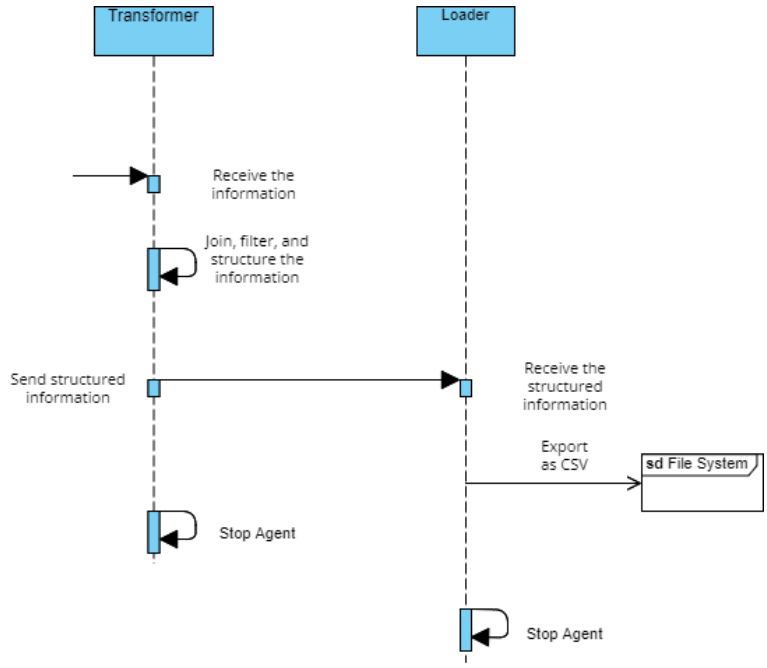

Figura 4.2: Diagrama de secuencia de la comunicación entre Agente Transformador y el de Carga

Como se ve en el título de la sección, el otro protocolo de comunicación del que debemos hablar es el de la comunicación que se produce entre el agente Transformador y el agente de Carga. Ya se ha explicado como el agente Transformador recibe la información y la organiza; bien, una vez que termina el proceso de filtrado y estructuración de los datos, llega el momento de que éstos sean enviados al agente de Carga para su exportación en forma de fichero *CSV*. Para ello, el mensaje tendrá la misma estructura que en el resto de mensajes que se enviaron anteriormente, es decir, el Dataframe será pasado a diccionario en forma de cadena, y será enviado una única vez al agente de Carga, que estará a la espera de este mensaje. Una vez que este mensaje haya sido enviado, el agente Transformador habra terminado todos sus comportamientos, y ejecución se dará por finalizada. Mientras tanto, cuando el agente de Carga haya recibido el mensaje, éste volvera a ser transformado de diccionario a Dataframe, para posteriormente exportarlo al directorio de trabajo en formato *CSV*, y finalmente, terminar su ejecución, y por ende la del programa.

Una vez que ya hemos explicado todo lo correspondiente a la parte técnica del proyecto, en el capítulo siguiente hablaremos de como poner en marcha el sistema.

CAPÍTULO 5

Puesta en marcha

En este capítulo, para finalizar, vamos a explicar como poner en marcha el sistema para su ejecución.

Este procedimiento es muy simple; para ello sólo necesitamos tener el ejecutable del proyecto, es decir, el archivo en formato *.py*, (código que se puede obtener del repositorio del proyecto), y tener *Python* instalado. De esta manera, tendremos que ejecutar el archivo como se ve en la imagen. Finalmente, tras la ejecución del programa, se obtendría un archivo *resultado.csv*, que contendría la información resultante del proceso ETL, y también un archivo llamado *log.log* con la información relativa a la traza de ejecución del programa.

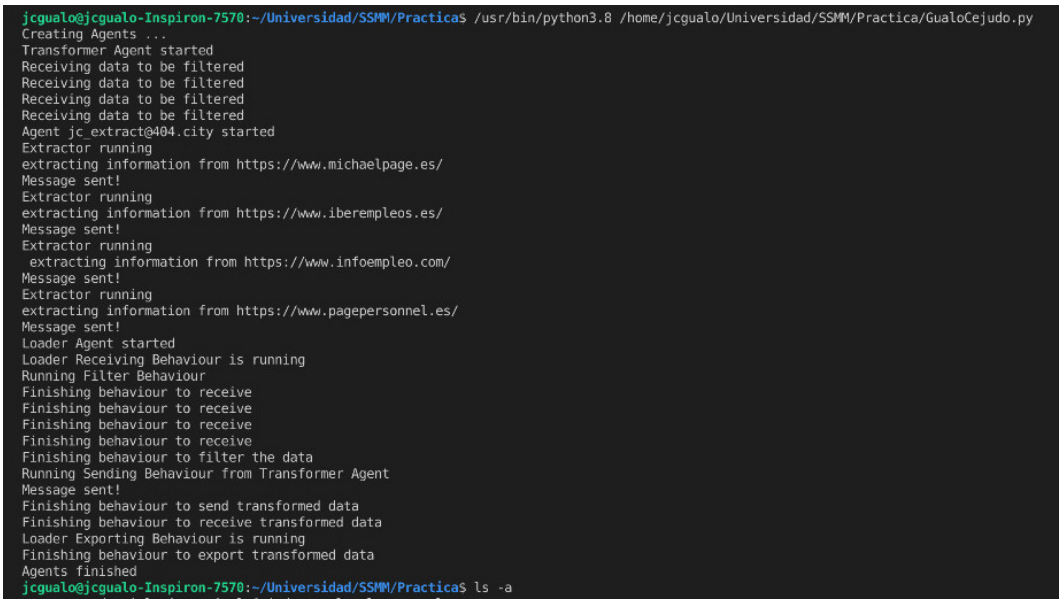

Figura 5.1: Ejemplo de ejecución del programa

CAPÍTULO 6

Enlaces de interés

En este apartado incluimos algunos enlaces que pueden ser de interés como son los siguientes:

(1) Enlace al repositorio de GitHub en el que se encuentra el código del proyecto, los archivos necesarios para ejecutarlo, y los archivos que genera una vez ejecutado.

https://github.com/JoseCarlosGualo/JoseCarlos_GualoCejudo_SSMM2021.git

(2) Enlace al video de Youtube en el que se muestra el programa en funcionamiento.

https://youtu.be/PjCe2k5os6Q