INTELIGENCIA ARTIFICIAL

E.T.S. de Ingenierías Informática y de Telecomunicación

Práctica 1

Agentes Conversacionales

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL
UNIVERSIDAD DE GRANADA
Curso 2018-2019

1. Objetivo

El objetivo de la primera práctica de la asignatura es familiarizarse con el diseño de agentes conversacionales, en particular con un lenguaje que ha sido especialmente diseñado para la descripción de bases de conocimiento en la construcción de agentes conversacionales, como es AIML.

Un agente conversacional es un agente software, diseñado con el objetivo de mejorar la comunicación entre los seres humanos y las máquinas, haciendo que estas últimas sean capaces de manejar conceptos del lenguaje natural (palabras, frases) como símbolos y reglas que actúan sobre estos símbolos.

Los agentes conversacionales tienen multitud de aplicaciones, siendo su principal labor en estos últimos años la de asistente para los seres humanos. Es muy conocido **SIRI**, incluido en los dispositivos desarrollados por la empresa ***Apple***. La relevancia de este tipo de agentes se pone de manifiesto en el creciente número de dispositivos y páginas de empresas que los incluyen, como por ejemplo, **GOOGLE NOW** una aplicación de características similares a SIRI desarrollada por ***Google*** para dispositivos Android, o **CORTANA** la aplicación desarrollada por ***Microsoft***, o los asistentes incluidos en las páginas web de empresas muy importantes como ***Ikea***. La propia ***Universidad de Granada*** tiene su asistente llamado **Elvira** que te ayuda a encontrar información relativa a la Universidad.

Muy recientemente, empresas como Google (con **Google Home**), Apple (con **HomePod**) o Amazon (con **Amazon Echo**) han sacado dispositivos empotrados en altavoces dedicados al control domótico mediante voz. Las versiones comerciales de estos dispositivos aún están en una fase muy inicial, y con ellos no se puede establecer una conversación como tal, son más bien modelos de pregunta y acción (como respuesta a la pregunta). Están empezando a poder establecer conversaciones cortas y ya se avanza en la línea de mantener conversaciones más parecidas a las que se esperaría entre seres humanos.

AIML no es el único lenguaje definido para el desarrollo de agentes conversacionales, ya que podemos encontrar otros como ***API.AI*** usado para el desarrollo de **Google Assistant** (una aplicación Android que incorpora una gran flexibilidad debida a su capacidad para aprender). Pero AIML es un lenguaje mejor desde el punto de vista docente y del que es más fácil encontrar información.

En el primer seminario de la asignatura se ha impartido un tutorial sobre el uso de este lenguaje, indicando cuáles son las sentencias más habituales e ilustrando su uso con algunos ejemplos y ejercicios. El **objetivo de esta primera práctica** consiste en definir bases de conocimiento en AIML para establecer algunas conversaciones sobre temas concretos, usando ***program-ab***, un intérprete para el lenguaje ***AIML 2.0*** de código abierto y que se puede descargar desde la página web de la asignatura (acceso identificado de ***decsai.ugr.es***).

2. Detalles de la práctica

Como se ha indicado anteriormente, la práctica está orientada al uso del lenguaje AIML y a mostrar sus posibilidades como medio para desarrollar agentes conversacionales. En concreto, se pide construir un

agente conversacional que sea capaz de seguir una conversación sobre dos posibles aplicaciones: (a) responder algunas preguntas habituales en las entrevistas de trabajo, y (b) desarrollar una aplicación que se pudiera incluir dentro dispositivos como *google home* que permitiera el manejo de plataformas de video bajo demanda. A continuación detallamos más cada ellas.

2.1. Agente conversacional para entrevistas de trabajo

Estaría muy bien que para acceder a un puesto de trabajo en lugar de tener que mandar un currículum y pasar una entrevista personal, pudieras enviar un bot que fuera capaz de exponer tu currículum de forma interactiva y además respondiera como lo haríamos nosotros a las preguntas de una entrevista de trabajo.

Así, el objetivo de esta parte es describir el conocimiento necesario para que un agente conversacional sea capaz de responder a las preguntas más habituales que se formulan en las entrevistas de trabajo, usando el lenguaje AIML y hacerlo de la manera que la haríamos nosotros.

Tanto las preguntas como una breve explicación de cómo debería ser la respuesta a dar en una entrevista real de trabajo se encuentran en el **Anexo 1**.

No sólo se pide que sepa responder a la pregunta en su formulación original, sino que sea capaz de responder correctamente usando variantes de la misma pregunta. Por ejemplo, si la pregunta original es "¿Dónde te ves dentro de 5 años?", debe también saber responder a otras dos variantes al menos, como por ejemplo "¿Dentro de 5 años donde crees que estarás?" o "¿En 5 años donde te ves?".

2.2. Agente Conversacional para manejar plataformas de streaming de contenidos multimedia bajo demanda por internet.

Uno de los problemas con los que actualmente se encuentran aplicaciones como **google assistant** implantado en los distintos dispositivos de **Google home** es que están muy limitados en el manejo de las aplicaciones asociadas para la visualización de contenidos visuales de video en streaming. **Google home** se pueden integrar con plataformas como netflix o HBO o los dispositivos **Echo** con la plataforma **Amazon Prime**.

El principal inconveniente de las integraciones anteriores es que los comandos verbales que admiten son extremadamente simples para permitir un control total de la aplicación exclusivamente por voz y obligan a usar el mando del televisor o teclado para la realización de las tareas algo más complejas. Pongamos como ejemplo, el comando asociado para ver una serie en *netflix* usando **Google home** sería: "*ok google, reproduce narcos en netflix*", esto empieza la reproducción del episodio de narcos en el punto donde dejamos de verlo.

Los comandos básicos disponibles son: pausar el video, continuar la reproducción del video, silenciar el video, activar el sonido del video, cerrar netflix.

Sin embargo, no están disponible comandos de voz que podrían ser interesantes para la visualización de sus contenidos como: Manejo de idiomas (poder cambiar el idioma del video), Poner subtítulos (poder activar y desactivar los subtítulos así como elegir el idioma de los subtítulos y el color o la posición de los mismo dentro de la pantalla), pasar al siguiente episodio, elegir un capítulo en concreto de una temporada, poner el episodio actual desde el principio, retrasar el video actual x minutos (rebobinar un poco el video), adelanta el video actual x minutos o reproduce el video actual a partir del minuto "x" (teniendo en cuenta que el valor de tiempo esté en el rango de duración del video).

En esta parte de la práctica, vamos a considerar tres etapas asociadas a niveles de desarrollo: (a) Introducción de los contenidos de las plataformas, (b) Inclusión de un conjunto de órdenes, claramente definidas, durante el proceso de visionado de los contenidos y (c) Inclusión de un conjunto adicional de órdenes que puedan ampliar la utilidad del sistema compuesto por los apartados (a) y (b).

Antes de hacer una descripción más detallada de las tres etapas anteriores, vamos a fijar el marco de trabajo en el que se va a desarrollar esta parte de la práctica.

2.2.1. Planteamiento del problema

Carlos y Marta son una pareja sin hijos, amantes exclusivamente de las series y que tienen contratados los servicios de dos plataformas “Netflix” y “Prime Video”. Ellos comparten su pasión por algunas series que visualizan juntos, pero sin embargo, también cada uno de ellos siguen series distintas que al otro no le interesan. Tienen en casa un dispositivo **Google Home** y desean usar comandos de voz para mejorar su experiencia en el visionado de las series de estas plataformas y usar lo menos posible el mando a distancia de la televisión.

En la realidad, Carlos y Marta no se tienen que preocupar de incluir la información relativa a los contenidos de la plataforma (eso ya lo hace la propia plataforma). Pero aquí vamos a diseñar al agente para que tenga la capacidad de incluir los contenidos en la plataforma mediante conversación. La información de partida de nuestro agente será la siguiente:

La plataforma “Netflix” ofrece las series “Narcos” y “La casa de papel”, mientras que de la “Prime Video” se tienen las series “The Expanse” y “Fleabag”. La información que se mantiene de cada serie es: género, idiomas tanto de audio como de subtítulos en los que se pueden reproducir, una corta descripción que indica de que va la serie, una dirección web donde se puede consultar más información sobre la serie, y por cada temporada, una secuencia con el nombre del capítulo y su duración expresada en minutos. Así, por ejemplo, la información relevante para la serie “Narcos” se describe en la siguiente tabla:

Género: drama, crimen, biografía Idiomas: español, inglés Subtítulos: español inglés Descripción: Narcos es una serie de ficción histórica basada en la lucha contra el narcotráfico en Colombia durante los años 1990

Página web: www.netflix.com/es-en/title/80025172

Temporada 1

- Descenso 57
- La espada de simon bolivar 47
- Los hombres de siempre 47
- El palacio en llamas 44
- Habra futuro 55
- Explosivos 50
- Lloraras lagrimas de sangre 51
- La gran mentira 51
- La catedral 51
- Despegue 45

Temporada 2

- Por fin libre 53
- Cambalache 47
- Nuestro hombre en madrid 47
- El bueno el malo y el muerto 56

La serie Narcos tiene 3 temporadas completas, cada una de ellas con 10 episodios. Sin embargo, en está práctica vamos a considerar que sólo tiene las dos primeras temporadas y de la segunda sólo están los 4 primeros episodios. La información completa de esta serie la podéis encontrar en www.netflix.com/es-en/title/80025172.

La información anterior de la serie Narcos, junto con la información considerada de las otras series, se puede encontrar en "**Descripcion_Series_Practica.txt**". Este fichero está incluido en el fichero comprimido que contiene el software necesario para la realización de la práctica.

A continuación describimos cada una de las etapas que han de ser consideradas en el desarrollo de la práctica.

2.2.2. Inclusión de nuevos contenidos en las plataformas

En esta primera fase, el sistema conversacional será utilizado para incluir nuevos contenidos en las plataformas anteriormente mencionadas. La inclusión de contenidos implica la realización de las siguientes acciones:

(1) Incluir un nuevo capítulo en una temporada de una serie.

(2) Incluir el primer capítulo de una nueva temporada en una serie.

(3) Incluir una nueva serie en una de las plataformas.

Para la realización de las tareas anteriores, es necesario definir una estructura y una codificación de la información que requiere cada serie.

En el Anexo 4, se muestra un ejemplo de diseño, así como se ilustra la forma en la que una nueva serie puede ser incluida en un catálogo. El diseño considerado en dicho Anexo puede ser usado por el estudiante, pero no es obligatorio. El estudiante puede decidir usar otra forma para representar la información.

2.2.3. Consultas sobre el contenido de las plataformas

En esta segunda fase, el sistema conversacional será utilizado para la realización de consultas sobre los contenidos de las plataformas. Deberá responder a preguntas semejantes a:

(1) Indicar los idiomas y subtítulos en los que se puede visionar una serie. Esto incluye preguntar si la serie está en un idioma concreto.

(2) Responder al número de idiomas, subtítulos o capítulos totales o de una temporada que tiene una serie en concreto.

(3) Indicar el número de temporadas que tiene la serie.

(4) Decir el nombre de un capítulo concreto de una temporada de una serie.

(5) Indicar la duración en minutos de un capítulo concreto de una temporada de una serie.

2.2.4. Control de reproducción de los contenidos

En esta tercera fase, incluiremos comandos que permitan facilitar el visionado de la series sin hacer uso del mando a distancia. En esta parte, hay que tener en cuenta el hecho de que hay dos usuarios, Carlos y Marta. Hay que tener en cuenta todas la casuísticas, es decir, ambos ven la misma serie por el mismo capítulo y punto de reproducción o ambos siguen la serie, pero cada uno a su ritmo. También considerar que uno sigue una serie que al otro no le interesa.

Dado que se pueden dar estas situaciones, es necesario definir un sistema de almacenamiento de la información que mantenga el estado de visionado de cada uno de ellos.

Así, al entrar en la fase de visionado en la plataforma, será necesario identificarse (soy Carlos, soy Marta o estamos los dos), y adaptar los comandos al usuario. Así, si siendo Marta se invoca al comando de "reproduce narcos", lo hará por el punto donde Marta dejó de ver la serie. Al igual que "reproduce", el resto de comandos afectarán exclusivamente al perfil de quién se ha identificado.

Además, cuando se entra en esta parte de visionado de contenidos, se inicializará una variable global llamada "tiempo", que irá reflejando el tiempo (siempre en minutos) que se está usando el sistema. Para simular que el tiempo avanza, se usará la acción "SIMUL minutos", que avanzará esa cantidad de minutos la variable global "tiempo". El sistema debe mantener la coherencia de la reproducción ante la invocación al comando "SIMUL", esto es, si el video está en reproducción, el punto de visualización del video avanzará la cantidad de tiempo indicada, pero si está pausado obviamente, no cambiará el punto de visionado. La definición preliminar de la acción "SIMUL" se encuentra en el archivo "simulatiempo.aiml". Esta acción puede ser modificada por el estudiante para considerar todos los posibles casos.

Las **habilidades básicas** que se le piden en esta parte de la práctica al agente son:

(1) Reproducir una serie

(2) Reproducir el capítulo actual a partir de un minuto concreto

(3) Retrasar/Adelantar la reproducción x minutos

(4) Detener la reproducción

(5) Cerrar la aplicación de visionado

(6) Activar/Desactivar los subtítulos, así como seleccionar el idioma

(7) Cambiar el audio entre los idiomas disponibles

En la respuesta de las 4 primeras acciones el agente debe devolver el punto temporal en el que se encuentra la serie en cuestión. En los dos últimos, debe contemplarse la posibilidad de qué hacer si el idioma pedido no está disponible.

Además, se le pide una **habilidad especial**. Esta habilidad es qué el agente entable una conversación para decidir que hacer cuando uno de ellos está viendo algo y llega el otro. Imaginad la siguiente situación: Carlos está viendo *Narcos* y llega Marta que no sigue esa serie. El agente podría proponerle detener *Narcos* en ese momento y pasar a ver *La casa de papel* si los dos van por el mismo punto de emisión o la diferencia entre sus puntos de emisión es pequeña. Estudiar aquí las distintas opciones y dejar que ellos decidan.

Un ejemplo ilustrativo de todo el proceso de simulación podría ser el siguiente:

Human: Reproduce la serie narcos
Robot: Ok, podrias decirme si eres Carlos o Marta
Human: Soy Carlos
Robot: Reproduciendo el palacio en llamas, el capítulo 4 de la primera temporada de Narcos por el minuto 30, donde lo dejaste la ultima vez

Human: SIMUL 10
Robot: El tiempo total transcurrido es de 10 minutos. Capitulo en reproduccion va por el minuto 40
Human: Retrocede 5 minutos
Robot: Capitulo en reproduccion va por el minuto 35
Human: SIMUL 10
Robot: El tiempo total transcurrido es de 20 minutos. Capitulo en reproduccion terminado
Human: Reproduce el siguiente capitulo
Robot: Reproduciendo habra futuro, el capítulo 5 de la primera temporada de Narcos por el minuto 0
Human: pon el idioma en ingles
Robot: Puesto el audio en la pista de ingles, capitulo en reproduccion va por el minuto 0
Human: Pon los subtitulos en frances
Robot: para la serie actual no tiene subtitulos en frances, puedes elegir entre espanol e ingles
Human: Da igual, no pongas subtitulos
Robot: ok, capitulo en reproducción va por el minuto 0
Human: SIMUL 16
Robot: El tiempo total transcurrido es de 36 minutos. Capitulo en reproduccion va por el minuto 16
Human: Ha llegado Marta
Robot: El capitulo 4 de la primera temporada de Fleabag lo dejasteis sin terminar, quereis que lo reproduzca ahora
Human: ahora no
Robot: El capitulo 9 de la primera temporada de la casa de papel lo estabais viendo los dos, lo reproduzco por el minuto que lo dejo Carlos
Human: vale
Robot: Reproduciendo episodio 9, el capítulo 9 de la primera temporada de la casa de papel por el minuto 13
Human: SIMUL 5
Robot: El tiempo total transcurrido es de 41 minutos. Capitulo en reproduccion va por el minuto 18
Human: Cerrar reproduccion
Robot: Aplicacion cerrada

3. Evaluación

3.1 Definición de los niveles de dificultad

Hemos diseñado un modelo de evaluación para que el alumno decida con qué intensidad y a qué nivel desea implicarse en su elaboración. Obviamente, a mayor nivel de implicación, el alumno opta a una mayor calificación en la práctica. Por tanto hemos definido 4 niveles en la entrega de esta primera práctica que son, de menor a mayor implicación, los siguientes:

(a) **Nivel 0**: Entrega del conocimiento necesario para responder a las preguntas sobre la entrevista de trabajo. En este caso, el alumno puede optar hasta cuatro puntos sobre diez.

(b) **Nivel 1**: Inclusión de los comandos necesarios para que el agente pueda incluir contenidos en las plataformas. Como mínimo deben considerarse los tres que se proponen en el apartado 2.2.2. En este caso, el alumno pude optar hasta una calificación de seis puntos sobre diez.

(c) **Nivel 2**: Inclusión de los comandos necesarios para que el agente pueda hacer consultas sobre los contenidos de las plataformas. Al menos deben hacerse los 5 que se consideran en el apartado 2.2.3. Para esta opción, el alumno puede obtener hasta una calificación de ocho sobre diez.

(d) **Nivel 3**: Inclusión de los comandos de control en la reproducción de las series. En este caso se tendrán que hacer al menos los 6 que aparecen como básicos en el apartado 2.2.4, incluyendo un control sin incoherencias del tiempo para obtener un nueve sobre diez. Para obtener el diez sobre diez, habrá que realizar la habilidad especial que también se sugiere en ese mismo apartado.

No se considerarán aquellas entregas que no puedan ubicarse en uno de estos niveles; es decir, estas son las únicas posibilidades de entrega. Para que un nivel pueda ser valorado, es necesario que se hayan superado todos los niveles anteriores.

Un ejemplo de entrega inválida, es aquella donde el alumno entrega el nivel 1 y no realiza el nivel 0, ya que esta última es obligatoria en todos los niveles.

3.2 ¿Cómo se evaluará cada uno de los problemas?

Adjunto a este guion, se establece una fecha límite para la entrega de la práctica. Antes de dicha fecha, el alumno debe subir el material que más adelante se indica conteniendo el conocimiento necesario para resolver los problemas hasta el nivel que ha decidido. En una segunda fecha posterior, que también se indica en este guión y que corresponderá con un día del horario habitual del grupo de prácticas se realizará la defensa de la práctica.

Hemos definido una forma de defensa para cada uno de los niveles definidos en la entrega. A continuación se indica cómo se evaluará cada nivel:

Nivel 0: En el anexo 1 de este guion se proponen 18 preguntas relativas a las preguntas más habituales en una entrevista de trabajo. El alumno debe definir el conocimiento necesario para responder correctamente a cada una de ellas. Además, se han definido otras preguntas que proponen respuestas semejantes a las que se proporcionan en las propuestas, pero que no están a disposición del alumno (básicamente, otras formas de preguntar lo mismo). Durante el proceso de defensa, teniendo en ejecución su agente conversacional, el alumno introducirá 2 preguntas que le indicará el profesor entre las 18 propuestas y otras 2 preguntas de entre las no conocidas. Para poder optar a tener puntuación en este nivel, el agente deberá responder correctamente a las 2 primeras preguntas, y superará este nivel si responde correctamente al menos a una de las no conocidas. En el caso de no responder correctamente ninguna de las 2 últimas preguntas, el profesor le pedirá al alumno que modifique su conocimiento en una de esas 2 preguntas y se verificará que lo ha hecho correctamente. En este mismo proceso, se verificarán las otras opciones que el estudiante había planteado para estas preguntas. Si lo hace correctamente, superará este nivel y obtendrá una calificación de 4. Si no lo sabe hacer, su calificación será de 2 puntos y la defensa termina.

En cualquier caso, el profesor siempre tiene la potestad de pedir al alumno que introduzca nuevo conocimiento o que le explique alguna parte de su código para asegurarse de que supera este nivel.

Nivel 1: Para llegar a este nivel en la defensa, debe haberse superado el nivel anterior. En este caso, la evaluación es simple: Se pedirá al estudiante que haga uso de las acciones de inclusión de

contenidos.

- El estudiante obtendrá 2 puntos más a su calificación, si realiza bien las tareas encomendadas por el profesor y su calificación será de 6 puntos,

En caso de una resolución no satisfactoria del nivel, la defensa termina y la calificación del alumno será de 4 puntos.

Nivel 2: Al igual que antes, debe haberse superado el nivel 1 con una puntuación de 6 puntos para pasar a evaluar este nivel, y del mismo modo la forma de evaluarlo es que el profesor propondrá consultas sobre la información contenida en la plataforma y el agente deberá responder con coherencia. Si el agente mantiene la conversación y la secuencia de respuestas son correctas, el alumno superará este nivel, y añadirá 2 puntos y su calificación será de 8. Si no es así, la defensa habrá terminado y la calificación del alumno será de 6 puntos.

Nivel 3: Para acceder a este nivel, el agente presentado para su evaluación ha debido superar el nivel 2 con una puntuación de 8 puntos. Para evaluar este nivel, se planteará una simulación en la que se empezará activando el sistema de visionado, se procederá a la identificación de el/los usuarios, para continuar con una secuencia de acciones que el agente debe hacer correctamente manteniendo de forma coherente tanto la evolución del tiempo global, como el avance de las series involucradas en el visionado. Si la simulación es satisfactoria, el estudiante añadirá un punto más a su valoración y se pasará a evaluar la habilidad especial. En otro caso, la evaluación termina y la nota del estudiante será de un 8.

Para la evaluación de la habilidad especial, se continuará con la simulación anterior y en algún punto se le pedirá que mantenga una conversación con los usuarios para determinar que hacer. El agente debe proporcionar a los usuarios una secuencia de alternativas inteligentes en función del estado de evolución de cada uno de ellos en las series que comparten su visionado. En este último caso, la valoración de este último apartado dependerá de la estrategia definida por el estudiante para resolver el conflicto, así que podrá tomar un valor entre 0 y 1 en función de las opciones contempladas por el agente.

3.3. ¿Qué hay que entregar?

Antes de que termine el plazo de entrega fijado en este guión, el alumno deberá subir a la plataforma de la asignatura en decsai.ugr.es, un archivo comprimido con zip, llamado “practica1.zip”. Este archivo debe tener la misma estructura en carpetas que cuelga de la carpeta “mybot”, donde obviamente las carpetas “aiml”, “aimlf“, “sets” y “maps” contienen los ficheros necesarios para resolver la práctica al nivel que ha decidido el alumno.

3.4. Observaciones Finales

Esta **práctica es INDIVIDUAL** y trata de establecer la capacidad del alumno para desarrollar una base de conocimiento usando el lenguaje AIML. El profesorado para asegurar la originalidad de cada una de las entregas, someterá a estas a un procedimiento de detección de copias.

En el caso de detectar prácticas copiadas, los involucrados (**tanto el que se copió como el que se ha dejado copiar**) tendrán suspensa la asignatura. Por esta razón, recomendamos que en ningún caso se intercambie código entre los alumnos. No servirá como justificación del parecido entre dos prácticas el argumento *"es que la hemos hecho juntos y por eso son tan parecidas"*, o "como estudiamos juntos, pues...", ya que como se ha dicho antes, **las prácticas son INDIVIDUALES**.

Como se ha comentado previamente, el objetivo de la defensa de prácticas es evaluar la capacidad del alumno para enfrentarse a este problema. Por consiguiente, se asume que todo el código que aparece en su práctica ha sido introducido por él por alguna razón y que dicho alumno domina perfectamente el código que entrega. Así, si durante cualquier momento del proceso de defensa el alumno no justifica adecuadamente algo de lo que aparece en su código, la práctica se considerará copiada y tendrá suspensa la asignatura. Por esta razón, aconsejamos que el alumno no incluya nada en su código que no sea capaz de explicar qué misión cumple dentro de su práctica y que revise el código con anterioridad a la defensa de prácticas.

Por último, las prácticas presentadas en tiempo y forma, pero no defendidas por el alumno, se considerarán como no entregadas y el alumno obtendrá la calificación de 0. El supuesto anterior se aplica a aquellas prácticas no involucradas en un proceso de copia. En este último caso, el alumno tendrá suspensa la asignatura.

4. Desarrollo temporal de la práctica

La práctica primera seguirá el siguiente desarrollo temporal:

a) **Del 21 de febrero al 27 de febrero**: Presentación de la práctica.

b) **Clases del 4 de marzo al 22 de marzo:** Desarrollo de la práctica en clase.

c) **Semana del 25 de marzo**: Defensa de la práctica que se realizará en el día y hora de la sesión de prácticas que le corresponde habitualmente al alumno. Si por algún motivo, el alumno no pudiera asistir a su sesión esa semana, debe avisar al profesor de prácticas para que le asigne a otro grupo de prácticas de la asignatura para esa misma semana.

d) **La fecha tope para la entrega** será **el domingo 24 de marzo antes de las 23:00 horas.**

NOTA: La semana del 11 de marzo se dedicará a resolver los ejercicios de la relación de problemas sobre agentes reactivos, por tanto, las sesiones de dicha semana no estarán dedicadas a resolver esta práctica.

ANEXO 1

Las 18 preguntas típicas en las entrevistas de trabajo

Aclaración: Se eliminan los símbolos de interrogación y las tildes para evitar problemas de codificación de caracteres en cada ordenador. Asociada a cada pregunta aparece una indicación de la forma en la que se debería responder (o en algún caso, como no deberías hacerlo). Esta información se ha obtenido de la página

"https://www.marketingandweb.es/emprendedores-2/preguntas-entrevista-trabajo/"

1. Hablame un poco de ti

Esta es una de las preguntas más clásicas y típicas en una entrevista, y que debemos saber responder pero de la manera más natural posible, y resumiendo en las menos palabras posibles nuestra experiencia profesional que aporta valor al puesto de trabajo que está ofertando la empresa.

2. Cual es tu mayor defecto

Rehúye las respuestas tópicas y típicas como; soy muy perfeccionista, soy muy exigente con mi trabajo y las personas, etc., lo mejor es que busques una respuesta más sencilla como; "A veces soy algo competitivo o me gusta trabajar de manera individual que en equipo".

3. Que aspectos no te gustan de un jefe

Atención, nos encontramos ante una pregunta trampa, que busca con ello añadir un criterio selectivo que descarte una serie de entrevistados, y a la que debemos tratar dar una respuesta del tipo como; "En mi experiencia profesional he trabajado con diferentes tipos de jefes y utilizando diferentes metodologías de trabajo, pero creo que soy una persona que me adapto bien a cualquier metodología de trabajo".

Tienes que pensar que en una entrevista de trabajo siempre hay varias preguntas trampa, que debemos identificar para poder dar una respuesta adecuada.

4. Cuales son tus 3 mayores virtudes

Sé conciso y evita explayarte en tu respuesta. El objetivo es indicar al entrevistador 3 aspectos concretos personales y profesionales que pueden ser positivos para el puesto de trabajo y la empresa.

Evita respuestas como; "Soy muy profesional y comprometido con mi trabajo", "Soy muy exigente y siempre doy muchísimo en todo lo que hago". Eso no se lo traga ningún entrevistador, al menos yo personalmente no me lo tragaría, y en lugar de estas respuestas utiliza otras como; "Me encanta aprender cosas nuevas y creo que en esta empresa puedo crecer mucho profesionalmente". Es decir, fíjate que lo que he hecho es mostrar una virtud y enlazarla con algo positivo mirando en el futuro de la empresa.

5. Te consideras preparado para desempeñar las funciones de este puesto

Tienes que estudiar muy bien cuáles son los puntos fuertes del perfil que está demandando la empresa y sobre todo ser sincero en la repuesta. Trata de recalcar los aspectos más fuertes que cumples del perfil solicitado y luego sé sincero y di que estos otros puntos los controlas a un nivel medio pero que te encantaría seguir aprendiendo y creciendo en esas áreas.

6. Tienes experiencia en la realizacion de proyectos en equipos de trabajo.

Un aspecto que suele valorar mucho en una entrevista de trabajo es nuestra experiencia y buena predisposición para trabajar en equipos de trabajo donde tengamos que interactuar con diferentes profesionales de la empresa. No seas exagerado y trata de dar una respuesta correcta y natural. "Sí, he trabajado en varios equipos de trabajo y con buenos resultados".

7. Realizarias un trabajo que no fuera con tu especialidad

El objetivo de esta pregunta es ver el grado de polivalencia del candidato, en función del puesto de trabajo a desempeñar puede ser un factor muy importante.

8. Te desplazarias de manera ocasional a otros lugares.

En este ocasión lo que se busca es ver el grado de movilidad del candidato, para ver si en caso de que tenga que acudir a una reunión de trabajo en otra ciudad su vida familiar se lo permitiría o no.

9. Te consideras una persona creativa

Piensa que la creatividad siempre aporta un plus a cualquier puesto de trabajo y debemos saber cómo responder correctamente a esta pregunta, eso sí siendo sincero, aquí no trates de ponerte adornos donde no los hay porque lo descubrirán relativamente muy pronto si no es cierto lo que dices. La creatividad es un aspecto muy personal y no tiene nada que ver con tener unos determinados estudios.

10. Que expectativas tienes de este trabajo

Cualquier persona antes de ir a una entrevista de trabajo ya tiene más o menos claro o dibujado algunas expectativas de ese nuevo trabajo. Estos son algunos de los aspectos que tendrías que destacar.

i. Me parece un puesto con mucha proyección para ...
ii. Considero que el puesto me permitirá crecer y aprender mucho.
iii. Realmente es un tema en el que me gusta trabajar y tengo experiencia.
iv. Estoy encantado con las funciones a desempeñar y me gustaría empezar ya.
v. Creo que es una empresa que cuida mucho que haya muy buen ambiente de trabajo entre sus empleados.

11. Que trabajos has tenido en esta tematica

Aquí se trata de que des una respuesta acorde al tema que te preguntan, es decir no se trata de que hablemos

sobre experiencias en otros trabajamos que no aportan nada a las funciones que tendría que desempeñar en la oferta de trabajo.

12. Normalmente te llevas el trabajo a casa

Es muy importante que establezcamos claramente la separación entre nuestro horario laboral con nuestra vida personal y privada, eso sí, puedes decir, no me importa si un día tengo que quedarme un poco más tarde mientras sea algo puntual o excepcional.

13. Cuales son tus aspiraciones salariales

Llegamos a la parte económica, y tienes que tener en cuenta que entramos en un tema delicado. Es posible que la oferta de trabajo que nos hayamos presentado hayan abierto un baremo muy amplio que nos genera confusión en lo que realmente nos van a ofrecer.

La empresa ya tiene establecido diferentes horquillas de precios en función de la experiencia y curriculum del candidato, pero tenemos que nosotros mismos tener claro cuánto es nuestra aspiración salarial para aceptar ese puesto de trabajo.

14. Donde te ves dentro de cinco años

Aquí no te vayas por las ramas y empieces a hablar por temas personales sino céntrate en sí te ves dentro de 5 años en la empresa y cómo te verías. “Yo dentro de 5 años me veo trabajando en la empresa y quizás coordinando algún área”.

15. No tienes experiencia para el puesto a desempeñar

Esto es algo muy habitual, especialmente cuando son jóvenes universitarios o personas que acaban de terminar los estudios, no tienen experiencia laboral, pero pese a eso es muy importante que destaques la habilidades más importantes que tienes, para que toda la importancia vaya ahí, y pierda importancia el hecho de que no tengamos experiencia.

16. Que sabes de la empresa y el puesto de trabajo

Cuando nos hemos interesados en conocer un poco la empresa y el puesto de trabajo que vamos a desempeñar, estaremos transmitiendo al entrevistador un gran interés y una buena actitud para trabajar en la empresa.

Una persona que no conoce nada o no ha tenido tiempo de mirarlo, transmite poco o nada de interés en el puesto de trabajo, y quizás no sea la mejor elección.

17. Que haces cuando consideras que tu jefe no tiene la razón

Aquí tenemos otra pregunta trampa a la que tenemos que saber dar una respuesta adecuada para salir del bache. Todos hemos tenido jefes que hacen cosas que no nos gustan pero lo que tenemos que tratar de buscar aquí es una respuesta prudente en lugar de sincera. Si somos demasiados sinceros el entrevistador podría confundir nuestra sinceridad con un persona problemática para la empresa.

18. Por que deberiamos escogerte a ti

Piensa que se presentan cientos de personas a una entrevista de empleo por lo que aquí tienes muy clara cuáles serán tus respuestas y que deberán ir enfocadas en aquellos aspectos diferenciadores y que hacen destacar entre los demás. “Llevo más de 5 años especializándome en esta área y creo que puedo aportar grandes cosas”. “La oferta de trabajo creo que se ajusta mucho a mi perfil”. “Me encantaría comenzar hoy mismo a trabajar en la empresa y ayudar a mis compañeros”. ¿Cual es la cualidad que tienes qué más valoras?

ANEXO 2

Durante la presentación de la práctica se ha hecho una descripción de las componentes más importantes del lenguaje AIML 2.0. En este anexo, se describen algunas funcionalidades adicionales que se han incluido en este lenguaje para aumentar su expresividad.

Extensión del lenguaje AIML

El lenguaje AIML 2.0 tiene algunas restricciones a la hora de poder trabajar con conjuntos y diccionarios, así también para poder utilizar patrones <pattern> con expresiones regulares. Es por ello que se ha desarrollado en un Trabajo Fin de Grado (realizado por Benjamín Vega Herrera) una versión extendida que contempla nuevas características que hacen más llevadera la programación en este lenguaje.

2.1 Conjuntos en AIML

Los conjuntos en AIML 2.0 se acceden mediante la etiqueta <set>, pero solamente se pueden utilizar dentro del <pattern> y no en el <template>, ya que se confundiría con la etiqueta <set> para guardar valores en variables. Para darle mayor flexibilidad se ha desarrollado una versión extendida del lenguaje que permite:

- Cargar en una variable los valores de un conjunto:

```
<set var="valores"><readset>telefono</readset></set>
```

- Guardar en un conjunto un nuevo valor:

```
<addtoset>
  <name>datos</name>
  <key>email</key>
</addtoset>
```

donde <name> hace referencia al nombre del fichero que almacena el "set", en este caso, el fichero se llama "datos.txt" y está en la carpeta "set", y <key> hace referencia al nuevo valor que se desea incluir, en este caso, "email". Es un conjunto, por tanto, si "email" ya existiera, no se incluye este valor de nuevo en el "set".

- Eliminar en un conjunto un valor:

```
<removefromset>
  <name>datos</name>
  <key>email</key>
</removefromset>
```

donde <name> hace referencia al nombre del fichero que almacena el "set", en este caso, el fichero se llama "datos.txt" y está en la carpeta "set", y <key> hace referencia al nuevo valor que se desea incluir, en este caso, "email".

2.2 Diccionarios en AIML

Los diccionarios en AIML 2.0 se acceden mediante la etiqueta <map>. Esta etiqueta puede utilizarse solamente dentro del <template>. Por ejemplo, puedo intentar guardar en una variable local llamada "eltel" el valor asociado al diccionario *telefono* utilizando *Sanchez* como clave de la siguiente manera (las dos formas son equivalentes):

```
<set var="eltel"><map name="telefono">Sanchez</map></set>
<set var="eltel"><map><name>telefono</name>Sanchez</map></set>
```

[1]

Lamentablemente los diccionarios en AIML 2.0 son fijos, es decir, no se pueden modificar desde dentro del AIML. Es por ello que se ha desarrollado una versión extendida del lenguaje que permite:

- Cargar en una variable las claves de un diccionario (así luego puedo averiguar si existe una cierta clave en un "map"):

```
<set var="claves"><readkeys>telefono</readkeys></set>
```

- Guardar en un diccionario un nuevo par (clave, valor):

```
<addtomap>
  <name>telefono</name>
  <key>Perez</key>
  <value>111222333</value>
```

1 Esta segunda versión resulta útil cuando el nombre del diccionario se encuentra almacenado en una variable.

```
</addtomap>
```

donde <name> hace referencia al nombre del fichero que almacena el "map", en este caso, el fichero se llama "telefono.txt" y está en la carpeta "map", <key> hace referencia a la nueva clave que se desea incluir, en este caso, "Perez", y <value> es el valor que se asocia con esa clave. Si la clave ya existe, esta operación no modifica el "map".

- Modificar el valor asociado a una clave de un diccionario:

```
<modifymap>
  <name>telefono</name>
  <key>Sanchez</key>
  <value>111222333</value>
</modifymap>
```

- Borrar del diccionario una par (clave, valor):

```
<removefrommap>
  <name>telefono</name>
  <key>Sanchez</key>
</removefrommap>
```

en este último caso, no es necesario indicar el campo <value>.

2.3 Nuevas opciones para <pattern>

Se ha agregado la posibilidad de utilizar algunas expresiones regulares más comunes dentro de la etiqueta <pattern>. A continuación se listan y ejemplifican cada una de ellas:

- **Palabras opcionales**: Muchas veces nos interesa poder permitir la aparición de una palabra opcional en una expresión regular (en otros lenguajes se utiliza el símbolo "?" seguido de lo que se quiere hacer opcional). En AIML se ha optado por incluir la palabra opcional entre paréntesis. Por ejemplo, el patrón

```
<pattern>tengo (muchas) posibilidades</pattern>
```

se enlazará correctamente con las frases: “tengo posibilidades” y con “tengo muchas posiblidades”. Incluso se pueden utilizar varias veces en una misma frase:

```
<pattern>un numero al azar (uno) (dos) (tres)</pattern>
```

se enlazará correctamente con “un numero al azar”, “un numero al azar uno”, “un numero al azar uno dos”, “un numero al azar uno dos tres”, “un numero al azar uno tres” y “un numero al azar dos tres”.

- **Subconjunto de palabras**: En algunos casos el uso del “*” es demasiado amplio, y nos gustaría poder restringirlo a un conjunto más pequeño. En AIML se ha optado por utilizar corchetes para indicar esto, de forma similar a otros lenguajes. Por ejemplo, si solo quiero permitir que puedan aparecer dos palabras (si o no) escribiría el siguiente patrón:

```
<pattern>el me dijo que [si no] queria</pattern>
```

esto permite que se enlace correctamente solo dos posibles frases: “el me dijo que si quería” o bien “el me dijo que no quería”.

- **Prefijos y sufijos**: Muchas veces nos resulta útil poder colocar la raíz de un verbo e indicar que todas las posibles conjugaciones también se redirijan a la misma porción de código. Para ello se ha incluido la posibilidad de utilizar sufijos mediante el carácter “+”.

```
<pattern>^ guard+ ^</pattern>
<template>Guardando dato</template>
```

De esta manera se puede llegar al trozo de código asociado a ese patrón cuando el usuario ingrese diferentes frases, ej: “por favor guardame este dato”, “no te olvides de guardar este dato”, “ya me estas guardando este dato”, etc. De forma similar se pueden utilizar prefijos anteponiendo el “+” a la palabra deseada.

```
<pattern>^ +ndo ^</pattern>

<template>Arreando que es gerundio</template>
```

ANEXO 3

Descripción del fichero "utilidades.aiml"

Dentro del software entregado para la realización de la práctica 1, se incluye un fichero llamado "utilidades.aiml" que incorpora algunas reglas genéricas que permitirán al alumno realizar con mayor comodidad su práctica.

En este anexo se describen las reglas incluidas en este fichero con algunos ejemplos para ilustrar su uso.

Las utilidades son las siguientes:

1. Operaciones para manejar una lista de nombres
2. Generar un número aleatorio en un rango
3. Otra forma de hacer un ciclo y comparar dos cadenas

Ahora describiremos cada una de ellas.

3.1. Operaciones para manejar lista de palabras

En AIML se trabaja exclusivamente con cadenas de caracteres y con procesamiento simbólico. Así, cada variable global o local que se usa es siempre de tipo cadena de caracteres. Por esa razón, es importante tener definidas funcionalidades dentro del lenguaje que faciliten el trabajo con este tipo de dato.

En el fichero "utilidades.aiml" hemos incluido algunas de las operaciones más frecuentes y que pueden tener uso para el desarrollo de los problemas que se piden en esta práctica. En concreto, se han incluido las siguientes operaciones:

- **TOP**: Dada una lista de palabras, devuelve la primera palabra de esa lista.
- **REMAIN**: Dada una lista de palabras, devuelve la misma lista de palabras quitando la primera palabra.
- **COUNT**: Dada una lista de palabras, devuelve el número de palabras que contiene.
- **FINDITEM [palabra] IN [listaPalabras]:** Determina si "palabra" es una palabra incluida en "listaPalabras".
- **SELECITEM [number] IN [listaPalabras]:** Devuelve la palabra que ocupa la posición "number" en "listaPalabras".
- **REMOVEITEM [number] IN [listaPalabras]:** Elimina de "listaPalabras" la palabra de posición "number".

- **ADDITEM [palabra] IN [listaPalabras]:** Añade "palabra" a principio "listaPalabras", sólo si "palabra" no estaba antes en "listaPalabras". En este segundo caso, la operación no hace nada.
- **CODE [listaPalabra]:** Transforma la lista de palabras en una sola palabra sustituyendo los espacios en blanco por "_". Si no hay espacios en blancos, esta función no cambia nada en el argumento.
- **DECODE [Palabra]:** Transforma la palabra en una lista de palabras, sustituyendo los "_" por los espacios en blanco. Si no hay "_", esta función no provoca ningún cambio.
- **DELETREA [Palabra]:** Construye una lista de palabras, donde cada palabra está formada por cada letra de la palabra que se pasa por argumento.

Este repertorio de operaciones no tiene sentido si se utiliza como salida directa del agente, sino que tiene como función procesar de una manera más elaborada la salida que se ofrecerá, y está concebido para trabajar sobre variables, ya sean globales (predicados) o locales.

Una aclaración antes de pasar a ilustrar su uso con algunos ejemplos: En las definiciones anteriores se ha hecho uso de conceptos como "palabra" y "lista de palabras" para matizar la intención con la que se han construido estas herramientas pero, en realidad, en todos los casos, los parámetros son cadenas de caracteres. Cuando se hace referencia a "palabra" se quiere indicar que es una secuencia de símbolos que está entre 2 separadores (entendiendo aquí separador por uno o varios espacios en blanco). Cuando se hace referencia a "lista de palabras", se indica que contiene una secuencia de palabras separadas por espacios en blanco.

Para ilustrar como se pueden usar, supongamos que existe una variable global llamada list_fruit que almacena una secuencia de frutas.

Ejemplo 1: Darle algunos valores a esa variable global mediante la definición de una regla.

Para esto, aprovecharemos la pregunta "¿Conoces algunas frutas?", para darle un valor inicial a esa variable.

```
<category>
<pattern> conoces algunas frutas</pattern>
<template>
    <think>
      <set name="list_fruit">fresa cereza naranja mandarina</set>
    </think>
    si, alguna conozco.
</template>
</category>
```

Ejemplo 2: Queremos ir actualizando esa variable global con nuevas frutas que nos pueda ir proporcionando el usuario.

Vamos a construir una regla qué aprenda nuevas frutas a través de afirmaciones del usuario del tipo "la manzana es una fruta" o "el membrillo es una fruta".

```
<category>
<pattern>la * es una fruta</pattern>
<template>
    <think>
      <set var="existe">
        <srai>FINDITEM <star/> IN <get name="list_fruit"/></srai>
      </set>
    </think>
    <condition var="existe">
      <li value="0">
        <think>
          <set name="list_fruit">
             <srai>
                ADDITEM <star/> IN <get name="list_fruit"/>
             </srai>
          </set>
       </think>
            Recordare que <star/> es una fruta.
        </li>
        <li>
            Ya sabia que <star/> es una fruta.
        </li>
    </condition>
</template>
</category>
```

El proceso de esta regla es bastante intuitivo,

1. Asigna en la variable local existe si lo que se pasa en <star/> es una de las frutas que ya conoce y se encuentran almacenadas en la variable global list_fruit. Para determinar la existencia de esa fruta hemos hecho uso de **FINDITEM**.
2. La variable existe almacena un 0, si no encuentra esa fruta en la lista y en ese caso lo que hace es añadirla al principio de la lista list_fruit, y ofrece el mensaje al interlocutor de que recordará esa fruta. Para añadir la nueva fruta hemos hecho uso de **ADDITEM**.
3. Si la variable existe almacena un valor distinto de 0, significa que en la posición almacenada por esa variable se encuentra la fruta. En este caso, no modifica la lista list_fruit, y lanza el mensaje de que ya sabía que eso era una fruta.

La regla anterior recoge el caso de un patrón del tipo *"la… es una fruta"*, pero nos gustaría recoger el caso también del patrón "el… es una fruta". La primera intención sería copiar esta regla, pegarla debajo en el editor y cambiar el "la" por un "el". Esto funcionaría, pero un experto programador en AIML se daría cuenta qué es más simple crear un nueva regla que invoque a la anterior y cambiando el nuevo patrón. El resultado sería el siguiente:

```
<category>
<pattern>el * es una fruta</pattern>
<template>
    <srai>la <star/> es una fruta</srai>
</template>
</category>
```

Como se puede observar <srai> hace un papel semejante al de la invocación de un método en un lenguaje de programación convencional, y aquí aprovechamos ese recurso.

Ejemplo 3: Ahora vamos a ampliar nuestro repertorio de reglas para permitir corregir algún error en nuestra lista de frutas. Supongamos que en un momento dado, por error, el interlocutor dijo: *"la mesa es una fruta"* y nuestro agente añadió *mesa* a la lista, pero el interlocutor poco después se da cuenta de su error y quiere corregirlo y nos dice *"fue un error, la mesa no es una fruta"*. Incluiremos una regla para permitir que esto se pueda hacer:

```
<category>
<pattern>no es una fruta la *</pattern>
<template>
    <think>
      <set var="pos">
        <srai>FINDITEM <star/> IN <get name="list_fruit"/></srai>
      </set>
    </think>
    <condition var="pos">
      <li value="0"> Como puedes pensar que considere eso una fruta
      </li>
      <li>
        <think>
          <set name="list_fruit">
           <srai>
             REMOVEITEM <get var="pos"/> IN <get name="list_fruit"/>
           </srai>
          </set>
       </think>
            Menos mal que me lo dices, yo creia que era una fruta.
        </li>
    </condition>
</template>
</category>
```

El proceso es semejante al que se ilustra en el *Ejemplo 2*, se busca si <star/> está en la lista de frutas mediante **FINDITEM** y almacenado su valor en pos. Si pos vale 0 entonces eso no se había considerado como fruta. En otro caso, el valor está dentro de la lista de frutas y hay que eliminarlo. Para eso usamos **REMOVEITEM** que elimina la palabra de posición pos de list_fruit y el resultado se reasigna a list_fruit.

De igual modo que en el ejemplo anterior, podemos incluir la regla que considera los patrones que son de la forma "… el… no es una fruta":

```
<category>
<pattern>no es una fruta el *</pattern>
<template>
    <srai> no es una fruta la <star/></srai>
</template>
</category>
```

Ejemplo 4: Ya podemos actualizar nuestra lista de frutas insertando y eliminando elementos de la misma, pero el interlocutor sólo nos puede dar la información para recordar la fruta de una en una. Sería interesante que pudiera darnos una lista de frutas que el agente fuera capaz de recordar del tipo "*la manzana, el platano, el melon y la ciruela son frutas*", donde aparecen delante de cada fruta un "el" o un "la" y antes de dar la última fruta aparece una "y". Obviamente, deberíamos usar las reglas definidas anteriormente que nos permiten modificar la lista.

```
<category>
<pattern> * son frutas</pattern>
<template>
    <think>
      <set var="lista"><star/></set>
      <set var="item">
        <srai>TOP <get var="lista"/></srai>
      </set>

    <condition var="item">
      <li value="y">
        <set var="item">
          <srai>SELECTITEM 3 IN <get var="lista"/></srai>
        </set>
        <srai> la <get var="item"/> es una fruta </srai>
      </li>

      <li value="la">
        <set var="lista">
          <srai>REMAIN <get var="lista"/></srai>
        </set>
        <set var="item">
          <srai>TOP <get var="lista"/></srai>
        </set>
        <loop/>
      </li>

      <li value="el">
        <set var="lista">
```

```
        <srai>REMAIN <get var="lista"/></srai>
      </set>
      <set var="item">
        <srai>TOP <get var="lista"/></srai>
      </set>
      <loop/>
    </li>

    <li>
      <srai> la <get var="item"/> es una fruta </srai>
      <set var="lista">
        <srai>REMAIN <get var="lista"/></srai>
      </set>
      <set var="item">
        <srai>TOP <get var="lista"/></srai>
      </set>
      <loop/>
    </li>
  </condition>
</think>
Recordare todas estas frutas
</template>
</category>
```

Vamos poco a poco:

1. La primero acción almacena el contenido de la parte variable del patrón en la variable local lista.
2. Se extrae la primera palabra de lista y se almacena en la variable ítem.
3. Ahora se plantea una estructura condicional en función del valor de ítem, que consideramos que puede tomar 4 valores diferentes: "y", "la", "el" u otro valor. En el caso de tomar el valor:
    a. "y", entonces estamos justo antes de terminar la secuencia de palabras. La siguiente palabra a "y" es un artículo que podemos ignorar y la siguiente es el nombre de la fruta. Por esa razón, usamos **SELECTITEM** para tomarla tercera palabra de la lista, que almacenamos en ítem, para después invocar a la regla que es de la forma *"la… es una fruta"*, dónde… es el valor de ítem. Esta es la última fruta de la lista, por tanto, se sale del ciclo. (por eso, a diferencia del resto de los casos, no termina con un <loop/>.
    b. "la" o "el", para las dos situaciones tengo que hacer lo mismo. En este caso, ignorar el artículo y pasar a evaluar la siguiente palabra que es la que contiene el nombre de la fruta. Esta operación se hace con la conjunción de **TOP** que extrae el primero de lo que queda en la lista y lo almacena en ítem, y **REMAIN** que devuelve la lista menos el primero, que se vuelve a almacenar en lista. Como en este caso la secuencia de palabras no ha terminado, se tiene que ciclar, por eso aparece <loop/>.

E.T.S. de Ingenierías Informática y de Telecomunicación. C/ Daniel Saucedo Aranda s/n. 18071. Granada - Tlf. +34 958 244019 - http://decsai.ugr.es

c. el caso por defecto. Si no es una "y", ni un "el", ni un "la", lo que tiene en ítem es el nombre de una fruta. En este caso, se invoca al igual que en el apartado (a) a la regla *"la… es una fruta"* que la añadirá si no existe en la lista, y pasa a la siguiente palabra de la misma forma que se indica en el apartado (b), es decir, con la conjunción de **TOP** y **REMAIN** y al igual que antes, quedan palabras por procesar, por tanto se cicla con <loop/>.

4. Terminado el ciclo el procesado de la cadena ha terminado y propone al interlocutor una frase en el sentido de que recordara las frutas.

En estos cuatro ejemplos hemos mostrado el uso de todas y cada una de las operaciones para el manejo de listas de palabras.

Las tres últimas operaciones son CODE, DECODE y DELETREA. Aquí pondremos un ejemplo de uso de cada una de ellas. Supongamos que hay frutas con nombre compuesto, por ejemplo, "nuez de macadamia" y queremos insertarla en nuestra lista de frutas. Si la insertamos tal cual, en los procesos descritos anteriormente de contrar, seleccionar, eliminar, etc. esta nueva única fruta va a ser considerada como si fueran tres "nuez", "de", "macadamia". Para resolver esto, podemos usar las funciones CODE y DECODE. La primera de ella, aplicada a la fruta anterior

```
<set var = "item" > </srai>
  <srai> CODE nuez de macadamia</srai>
</set>
```

nos devuelve en la variable local "item" el valor "*nuez_de_macadamia*" como una única palabra.

La operación contraria es DECODE que dada una palabra que representa una lista de palabras pero separadas en lugar de espacios en blanco por caracteres "_" la trasforma en una verdadera lista de palabras. Así,

```
<set var = "item" > </srai>
  <srai> DECODE nuez_de_macadamia</srai>
</set>
```

devuelve en la variable "item" la cadena "*nuez de macadamia*".

Por último, DELETREA como su nombre indica, transforma una palabra o lista de palabras en una lista de palabras formada por las letras que componen la palabra pasada como argumento.

El resultado de

```
<set var = "item" > </srai>
  <srai> DELETREA nuez de macadamia</srai>
</set>
```

es que la variable "item" contiene la secuencia "*n u e z d e m a c a d a m i a*".

3.2. RANDOM

La sintaxis de esta acción es **RANDOM [number]** y devuelve un número aleatorio entre 1 y number. Aquí mostramos un ejemplo de uso de RANDOM.

Ejemplo 5: Se plantea una regla simple que ante la entrada de "Dime una fruta", el agente conversacional devuelve aleatoriamente una de entre la lista de frutas que tiene almacenadas en la variable list_fruit.

La descripción de dicha regla sería la siguiente:

```
<category>
<pattern> Dime una fruta </pattern>
<template>
   <think>
     <set var="lista"> <get var="list_fruit"/> </set>
     <set var="cantidad"><srai>COUNT <get var="lista"/></srai></set>
     <set var="pos"><srai>RANDOM <get var="cantidad"/></srai></set>
     <set var="elegida">
          <srai>
            SELECTITEM <get var="pos"/> IN <get var="lista"/>
          </srai>
     </set>
   </think>
   <get var="elegida"/>
</template>
</category>
```

El proceso de respuesta a la pregunta sería el siguiente:

1. Asigna a la variable lista una secuencia de nombres de frutas.
2. Mediante **COUNT** determina el número de frutas introducidas en lista y se la asigna a la variable cantidad.
3. A partir de cantidad, qué indica el número de frutas elegibles, usa **RANDOM** para generar un número aleatorio entre 1 y cantidad que se almacena en pos.
4. Selecciona la palabra que ocupa la posición pos de lista y la almacena en elegida usando **SELECTITEM.**
5. Devuelve como respuesta el valor de la variable elegida.

3.3. ITERATE / NEXT y COMPARE

Con el par ITERATE/NEXT tratamos de hacer una versión más convencional de un ciclo que itera sobre una lista de palabras. El proceso es simple: ITERATE se usa sólo una vez al principio del ciclo, y permite situarse sobre la primera palabra de la lista de palabras. El resto del proceso está guiado por NEXT, que devuelve el siguiente valor de la lista. Tanto

ITERATE como NEXT devuelve la cadena “end” cuando se termina de recorren la lista de palabras.

Ejemplo 6: Construir una regla que devuelva todas las frutas que conoce el agente.

En principio, esta regla sería tan simple como devolver el valor de la variable list_fruit, pero nosotros vamos a complicarlo un poco para usar estas acciones. Una posible implementación es la siguiente:

```
<category>
<pattern> Dime todas las frutas que conoces </pattern>
<template>
   Estas son las frutas que conozco
   <think>
     <set var=”item”>
        <srai> ITERATE <get name=”list_fruit”/> </srai>
     </set>
   </think>
     <condition var=”item”>
        <li value=”end”></li>
        <li> <get var=”item”/>
               <think>
               <set var=”item”>
                 <srai>NEXT</srai>
               </set>
             </think>
             <loop/>
        </li>
     </condition>
</template>
</category>
```

Creemos que este último ejemplo es fácil de seguir, si se ha entendido todo lo que se ha descrito anteriormente en este Anexo 3, así que dejaremos que lo intentéis por vosotros mismos.

ANEXO 4

Ejemplo para empezar la construcción del asistente para las plataformas de video bajo demanda.

A4.1 Definiendo una posible representación de la información

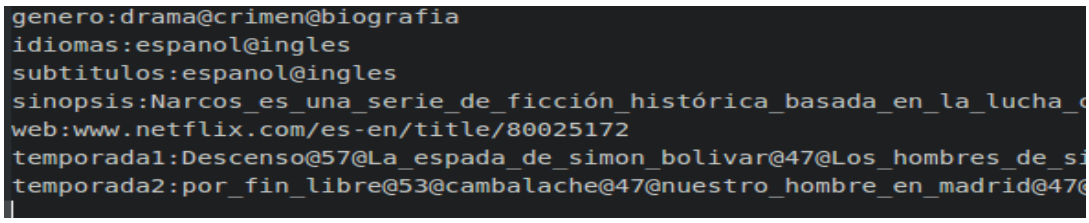
Empezaremos definiendo las estructuras de datos que necesitamos para mantener la información de la plataforma. En estas estructuras de datos se encuentra la información sobre los contenidos que ofrece cada una de las plataformas, así como las variables necesarias para las funcionalidades que se desean implantar.

El lenguaje AIML únicamente puede trabajar de forma nativa con "SETS", "MAPS" para almacenar la información entre ejecuciones, y variables globales para pasar información entre reglas en tiempo de ejecución. Así que para satisfacer esta parte de la práctica, es necesario definir los sets y maps que se van a usar.

Empezaremos con una definición inicial de la estructura, para ir modificándola conforme vayamos incluyendo funcionalidades. En esta estructura inicial, vamos a considerar

1. Un sets llamado "seriesnetflix" que contendrá la lista de series que ofrece ese operador. Inicialmente contendrá únicamente la serie "narcos".
2. Un maps llamado "narcos", el mismo nombre que utilizo en el set anterior que contendrá las siguientes claves: **genero**, **idiomas**, **subtítulos**, **sinopsis**, **web**, **temporada1**, **temporada2**,…

En la parte de valor, pondremos la siguiente codificación. El símbolo "_" representa al espacio en blanco y el símbolo "@" representa la separación entre dos valores posibles distintos para esa clave.
Un ejemplo de cómo quedaría ese map se muestra en la siguiente figura:



Se puede observar, que en el caso de las temporadas, el campo valor representa las secuencia de capítulos de la temporada, dónde el orden indica el ordinal del capítulo y junto al nombre se añade la duración del mismo en minutos, usando el símbolo '@' como separador.

A4.2 Haciendo una consulta simple sobre la estructura

Teniendo únicamente la estructura definida en el apartado anterior, vamos a intentar construir en AIML las reglas necesarias para sacar información sobre esta serie. Por ejemplo, sacar la sinopsis de una serie a partir de su nombre.

Vamos a empezar con la regla básica que sería:

```
<category>
  <pattern> De que va la serie * </pattern>
  <template>
          <map><name><star/></name>sinopsis</map>
  </template>
</category>
```

Si ejecutamos el intérprete y le preguntamos "De que va la serie narcos", nos dirá "Narcos_es_serie_de_ficcion_...", es decir, aparece el símbolo "_" en lugar del espacio en blanco. Para corregir lo anterior, en el fichero "utilidades.txt" aparece una regla llamada "decode", que transforma los símbolos anteriores por espacios en blanco. Incluyendo la llamada a esta regla, la regla anterior quedaría como:

```
<category>
  <pattern> De que va la serie * </pattern>
  <template>
          <srai>decode <map><name><star/></name>sinopsis</map></srai>
  </template>
</category>
```

Pero qué ocurre si el nombre de la serie está formado por más de una palabra, como por ejemplo "La casa de papel". Para probarlo, añadimos en el fichero "seriesneflix.txt" de la carpeta sets "la casa de papel", y creamos un nuevo map en la carpeta maps llamada "la casa de papel.txt", donde metemos la información sobre la serie. En la siguiente figura aparece como quedaría este fichero.

```
genero:drama@crimen@suspense
idiomas:frances@espanol@ingles
subtitulos:espanol@ingles@frances@ruso@chino@japones
sinopsis:Ocho_atracadores_toman_rehenes_en_la_fabrica_nacional_de_
web:www.netflix.com/es/title/80192098
temporada1:Episodio_1@80@episodio_2@67@episodio_3@63@episodio_4@64
temporada2:episodio_1@63
```

Ahora, desde el intérprete de AIML, preguntamos "de que va la serie la casa de papel" y nos responderá "Ocho atracadores toman…". Por lo tanto, para añadir una nueva serie, tenemos que crear un map con la información de la serie y añadir en el fichero "seriesnetflix.txt" el nombre de la serie. Como las series de las que tenemos información están incluidas en ese fichero, podemos adaptar la regla anterior de la siguiente forma:

```
<category>
  <pattern> De que va (la serie) <set>seriesnetflix</set> </pattern>
  <template>
          <srai>decode <map><name><star/></name>sinopsis</map></srai>
  </template>
</category>

<category>
  <pattern> De que va (la serie) * </pattern>
  <template>
La serie <star/> no está en el catálogo.
  </template>
</category>
```

Se han creado dos reglas que tiene la misma parte fija del patrón, pero en la primera se usa un set y en la segunda el asterisco. La primera tiene prioridad sobre la segunda, por tanto, si la parte variable coincide con un valor contenido en el fichero "seriesnetflix.txt" se disparará. En otro caso, será la segunda la que se dispare indicando que la serie no está en el catálogo. También se ha añadido un paréntesis "(la serie)" para poder omitirla en la pregunta.

Supongamos ahora que queremos que el asistente detecte que la serie no está incluida dentro del catálogo y la queremos incluir. ¿Cómo podemos hacer eso? Observemos primero que ocurre cuando le pedimos la sinopsis de una serie que no está. Preguntamos "*de que va la serie iron fist*" y nos devuelve "*La serie iron fist no esta en el catalogo*", es decir, se ha disparado la segunda regla. En este punto, si el usuario quiere, podríamos describirle los pasos para incluir la serie en el catálogo. Hay pasos que debe hacer el usuario fuera del interprete, en este caso, crear el archivo "*iron fist.txt*" en la carpeta map, y el resto, el usuario lo puede hacer fuera o hacerlo el intérprete si responde a los datos que le pide el intérprete. Un ejemplo de conversación podría ser el siguiente:

Human: de que va iron fist
Robot: La serie iron fist no esta en el catalogo. Quieres añadirla?
Human: Si, incluyela en el catalogo de netflix

Robot: ok, crea en la carpeta maps un archivo llamado iron fist.txt. Cuando hayas terminado reiniciame y al volver, pideme que anada la serie iron fist.

---------- Después de reiniciar el interprete ----------

Human: anade la serie iron fist
Robot: perfecto, te voy a preguntar datos sobre la serie iron fist. Indicame que plataforma la ofrece
Human: netflix
Robot: a que genero pertenece
Human: ciencia ficcion
Robot: de que va la serie
Human: Danny Rand reaparece 15 años despues de ser dado por muerto. Ahora, armado con un increible poder, tratara de recuperar su pasado y cumplir su destino.
Robot: dime la lista de idiomas en los que está disponible el audio de la serie
Human: español e ingles
Robot: dime la lista de subtitulos en los que está disponible la serie
Human: espanol, ingles y frances
Robot: si conoces una direccion web de la serie indicamela
Human: no la se
Robot: ya he creado la informacion basica de la serie, si quieres añadir capitulos me lo dices
Human: ahora no
Robot: ok

Esta conversación no es más que un ejemplo para ilustrar una forma de resolver el problema.

A4.3 Cómo hacer algo parecido a un "cin >> titulo"

Una de las cosas que más choca al enfrentarse a AIML es la forma en la que se pueden capturar entradas de información bajo petición del asistente. Cuando uno piensa en entradas de datos en un lenguaje como C++, como por ejemplo, pedir el título de un capítulo y su duración, podría imaginar un código semejante al siguiente:

```
cout << "Introduce el titulo del capitulo: ";
cin >> titcap;
cout << "Introduce la duracion: ";
cin >> durcap;
```

Pero, ¿cómo se puede hacer esto en AIML? La primera idea podría ser generar una regla semejante a la siguiente intentando emular el trozo de código anterior en C++:

```
<category>
  <pattern> Introduce un nuevo capitulo de la temporada <set>number</set> de * </pattern>
  <template>
        - - - - - - - - - - - - - - - -
           Dime el titulo del capitulo
          <set var="titcap"><srai>INTRODUCIRTITULO</srai></set>
          Dime la duración del capitulo
          <set var="durcap"><srai>INTRODUCIRDURACION</srai></set>
            - - - - - - - - - - - - - - - -
  </template>
```

```
</category>
```

Donde INTRODUCIRTITULO e INTRODUCIRDURACION son dos reglas que permiten capturar la información que les proporciona el humano.

Esto **NO FUNCIONA en AIML**. El mecanismo de funcionamiento del interprete es (1) el humano dice algo, (2) ese algo dispara una regla de la base de conocimiento que toma el control, (3) la regla no devuelve el control para que el humano diga algo hasta que termine de ejecutarse. En el ejemplo anterior, el humano dice "Introduce un nuevo capitulo de la temporada 2 de narcos", y ya no puede decir nada más hasta que la regla anterior termine. Pero para que la regla anterior termine, el humano tiene que decir el título del capítulo, pero como la regla no ha terminado el humano no puede decir nada.

Vale, ya sabemos que no se hace así. ¿Entonces como se hace? Bueno, pues está claro que la entrada debe producirse entre al menos dos reglas. Esto es, una primera regla termina pidiendo al humano que introduzca el título, la siguiente regla toma el título, lo almacena de alguna manera y termina pidiendo la duración, y la tercera y última regla toma la duración y realiza la tarea de almacenarlo en la estructura.

Pensemos en la siguiente estructura de conversación para insertar título de capítulo y duración:

Human: Introduce un nuevo capitulo de la temporada 2 de narcos
Robot: Dime el titulo del capitulo
Human: Los enemigos de mi enemigo
Robot: Dime la duracion del capitulo
Human: 52
Robot: Este capitulo ya esta disponible en la plataforma

Así, por cada intervención del humano hay que construir una regla que se dispare con las entradas esperadas. El esquema de las tres reglas podría ser algo como:

```
<!-- Regla 1 -->
<category>
  <pattern> Introduce un nuevo capitulo de la temporada <set>number</set> de * </pattern>
  <template>
        - - - - - - - - - - - - - - -
           Dime el titulo del capitulo
  </template>
</category>

<!-- Regla 2 -->
<category>
  - - - - - - - - - - - - - - -
  <template>
        - - - - - - - - - - - - - - -
           Dime el titulo del capitulo
  </template>
</category>


<!-- Regla 3 -->
```

```
<category>
  - - - - - - - - - - - - - - -
  <template>
        - - - - - - - - - - - - - - -
           Este capitulo ya esta disponible en la plataforma
  </template>
</category>
```

Bueno, ya sabemos que la petición tiene que ser dividida en este caso en tres reglas semejantes a las anteriores, pero no basta sólo con eso, ya que queremos que si se dispara la regla 1, la siguiente regla que se dispare sea la regla 2 y después de disparar la regla 3 tiene que dispararse obligatoriamente la regla 3. Por tanto, tenemos que indicar de alguna forma en el conocimiento que se tiene que hacer esa secuencia de reglas.

El lenguaje nos ofrece dos posibilidades relacionadas con lo que ya se vio de trabajar con el contexto de la conversación cuando se describió el lenguaje. Estas dos opciones son (1) usar “topic” o (2) usar el tag “<that>”. Veamos cómo se resuelve en cada caso.

A4.3.1 Resolución usando tópicos

Tópicos, “topic” como palabra reservada del lenguaje, permite dividir el conocimiento incluido en el lenguaje para que el proceso de búsqueda se restrinja a un conjunto específico de reglas. Por defecto, topic tiene el valor “unknown”, y por tanto busca entre todas aquellas que no están dentro de ningún tópico. Cuando se fija tópico a un valor concreto, entonces sólo se buscará sobre las reglas que están en ese “topic”.

La solución siguiendo esta estrategia podría ser:

```
<category>
<pattern>Anade un nuevo capitulo de la temporada <set>number</set> de * </pattern>
<template>
    <!-- Operaciones para verificar que existe lo que se pide -->
    <set name="serie"><star index="2"/></set>
    <set name="temporada"><star index="1"/></set>
    <set name="topic">titcap</set>
     Dime el titulo del capitulo
</template>
</category>

<topic name ="titcap">

<!-- Pide el nombre de un capitulo -->
<category>
 <pattern> * </pattern>
 <template>
     <set name="titulo"><star/></set>
     <set name="topic">durcap</set>
   Dime la duracion del capitulo
 </template>
</category>

</topic>
```

```
22. <topic name ="durcap">

23. <!-- Pide la duracion de un capitulo -->
24. <category>
25.  <pattern><set>number</set></pattern>
26.  <template>
27.      <set name="duracion"><star/></set>
28.      <set name="topic">unknown</set>
29.    <!-- Operaciones previas a la insercion en el map -->
30.      <modifymap>. . . . . </modifymap>
31.      Este capitulo ya esta disponible en la plataforma
32.  </template>
33. </category>

34. </topic>
```

De la línea 1 a la 10 aparece la primera de las reglas que lanza el resto de la secuencia. Las líneas 5 y 6 declaran dos variables globales "serie" y "temporada" que almacenan la información de serie y temporada del capítulo a insertar. Se almacenan en variables globales para ser accesibles en la tercera de las reglas de la secuencia qué será la encargada de pasar la información a la estructura. La línea 7 es la que se encarga de cambiar el tópico donde realizará la búsqueda el agente y se fija a "titcap". Por último, antes de terminar la regla devuelve como respuesta el mensaje "Dime el titulo del capitulo".

De la línea 11 a la 21 se describe el ámbito del tópico "titcap". Este tópico está formado por una única regla (líneas de la 13 a la 20). El patrón es "*", es decir, que cualquier cosa que escriba el interlocutor se considerará para disparar esta regla y eso que escriba será directamente el título del capítulo. En la línea 16, se crea la variable global "titulo" y se almacena en ella el título. En la línea 17, se cambia el tópico a "durcap". Por último, se ofrece como respuesta el mensaje "Dime la duracion del capitulo".

De la línea 22 a la 34 se describe el ámbito del tópico "durcap". También este tópico está formado por una única regla que además es la última regla de la secuencia. El patrón (línea 25) pide que se introduzca algo que debe ser un número. En la línea 27 se almacena en una variable global llamada "duracion", lo que se recibió como duración del capítulo, así, las variables globales "serie", "temporada", "titulo" y "duracion" configuran toda la información necesaria para insertar el nuevo capítulo en el map (si se sigue la organización de la información que se propone en los apartados anteriores). La línea 28 fija el tópico por defecto, de manera que desaparece la restricción en el proceso de búsqueda y ahora ya cualquier regla puede ser disparada. El comentario incluido en la línea 29 debe ser sustituido por la secuencia de tags en AIML que permiten componer la solución en el formato necesario, mientras que la línea 30, cambia el valor anterior en el map por el que haya compuesto. La regla termina respondiendo "Este capitulo ya esta disponible en la plataforma".

A4.3.2 Resolución usando el tag <that>

Esta segunda forma de resolución permite enlazar las reglas 2 y 3 a partir de lo que el agente responde al humano en las reglas 1 y 2. Se sigue manteniendo el uso de variables globales para que la información completa necesaria para insertar el nuevo capítulo llegue a la regla 3.

La solución usando "that" quedaría de la siguiente forma:

```
1. <category>
2. <pattern>Anade un nuevo capitulo de la temporada <set>number</set> de * </pattern>
```

```
<template>
    <!-- Operaciones para verificar que existe lo que se pide -->
    <set name="serie"><star index="2"/></set>
    <set name="temporada"><star index="1"/></set>
     Dime el titulo del capitulo
</template>
</category>

<!-- Pide el nombre de un capitulo -->
<category>
 <pattern> * </pattern>
 <that>Dime el titulo del capitulo</that>
 <template>
     <set name="titulo"><star/></set>
   Dime la duracion del capitulo
 </template>
</category>

<!-- Pide la duracion de un capitulo -->
<category>
 <pattern><set>number</set></pattern>
 <that>Dime la duracion del capitulo</that>
 <template>
     <set name="duracion"><star/></set>
   <!-- Operaciones previas a la insercion en el map -->
     <modifymap>. . . . . </modifymap>
     Este capitulo ya esta disponible en la plataforma
 </template>
</category>
```

A4.4 Ejemplos de consultas para el nivel 2

Para simplificar la construcción de las reglas que realizan las acciones anteriores, sería interesante definir *sets* que restrinjan los valores de los distintos campos. Por ejemplo, un *sets* que se llame "*idiomas.txt*" y que incluya la lista de idiomas posibles y lo mismo se puede hacer para género.

A modo de ejemplo, resolveremos la parte del punto (1) en relación a los idiomas disponibles de audio de una serie. Partiremos de la siguiente regla

```
<category>
  <pattern> En que idiomas esta (la serie) <set>seriesnetflix</set> </pattern>
  <template>
          <map><name><star/></name>idiomas</map>
  </template>
</category>
```

La regla anterior, ante la pregunta "en que idioma esta narcos", nos responderá "español@ingles". En realidad no debería responder "esta en español y ingles", eliminando el símbolo '@' por un espacio en blanco. Aunque en correcto castellano debería aparece ".. e inglés", admitiremos el uso de la 'y' en todos los casos (por supuesto, quién quiera puede hacer la transformación en la expresión correcta en castellano). Ya vimos antes que en utilidades aparece una regla llamada "decode" que cambia "_" por un espacio en blanco. Lo que vamos a hacer, es incluir la descripción de dicha regla en el nuevo fichero y cambiaremos ese símbolo por el '@'. El resultado sería la nueva regla "decode_fields":

```
<category>
   <pattern> decode_fields * </pattern>
   <template>
   <think>
        <set var="palabra"></set>
        <set var="solucion"></set>
        <set var="tmp"><star/></set>
        <set var="tmp"> <explode><get var="tmp"/></explode> </set>
        <set var="caracter"><first><get var="tmp"/></first></set>
        <set var="tmp"><rest><get var="tmp"/></rest></set>
        <condition var="caracter">
                <li value="NIL">
                        <set var="solucion"><get var="solucion"/> <get var="palabra"/></set>
                </li>
                <li value="@">
                        <set var="solucion"><get var="solucion"/> <get var="palabra"/></set>
                        <set var="palabra"></set>
                        <set var="caracter"><first><get var="tmp"/></first></set>
                        <set var="tmp"><rest><get var="tmp"/></rest></set>
                        <loop/>
                </li>
                <li>
                        <set var="palabra"><get var="palabra"/><get var="caracter"/></set>
                        <set var="caracter"><first><get var="tmp"/></first></set>
                        <set var="tmp"><rest><get var="tmp"/></rest></set>
                        <loop/>
                </li>
        </condition>
  </think>
  <get var="solucion"/>
  </template>
</category>
```

Usando la regla anterior, podemos adaptar la regla que pregunta por la lista de idiomas como:

```
  <category>
    <pattern> En que idiomas esta (la serie) <set>seriesnetflix</set> </pattern>
    <template>
       <think>
          <set var="tmp"><map><name><star/></name>idiomas</map></set>
          </think>
          <srai>decode_fields <get var="tmp"/></srai>
    </template>
  </category>
```

Dando como resultado ahora “espanol ingles”. Ahora quedaría añadir un “y” entre los dos últimos idiomas. Para ello, podríamos definir una nueva regla que pusiera una “y” entre las dos últimas palabras de una lista de palabras. La idea sería, contar el número de palabras de la lista, y concatenar “ y “ antes de incluir la última palabra. Esta regla podría quedar así:

```
    <category>
```

```
    <pattern> ANIADEY * </pattern>
    <template>
      <think>
      <set var="lista"><star/></set>
      <set var="tamano"><srai>count <get var="lista"/></srai></set>
      <set var="solucion"></set>
      <condition var="tamano">
        <li value="0"></li>
        <li value="1">
          <set var="solucion"><get var="lista"/></set>
        </li>
        <li>
          <condition var="tamano">
            <li value="1">
              <set var="solucion"><get var="solucion"/> y <get var="lista"/></set>
            </li>
            <li>
              <set var="palabra"><first><get var="lista"/></first></set>
              <set var="lista"><rest><get var="lista"/></rest></set>
              <set var="solucion"><get var="solucion"/> <get var="palabra"/></set>
              <set var="tamano"><map name="predecessor"><get var="tamano"/></map></set>
              <loop/>
            </li>
          </condition>
        </li>
      </condition>
      </think>
      <get var="solucion"/>
    </template>
  </category>
```

y añadiendo esta nueva regla a la definición anterior, tendríamos:

```
  <category>
    <pattern> En que idiomas esta (la serie) <set>seriesnetflix</set> </pattern>
    <template>
       <think>
          <set var="tmp"><map><name><star/></name>idiomas</map></set>
          </think>
          <srai>aniadey <srai>decode_fields <get var="tmp"/></srai></srai>
    </template>
  </category>
```

Ahora, si la pregunta es si una serie está en un idioma concreto, se podría hacer de la siguiente forma:

```
<category>
    <pattern> ^ <set>seriesnetflix</set> esta en <set>idioma</set>  </pattern>
    <template>
        <think>
          <set var="lista"><map><name><star index="2"/></name>idiomas</map></set>
          <set var="lista"><srai>decode_fields <get var="lista"/></srai></set>
```

```
        <set var="idioma"><first><get var="lista"/></first></set>

        <condition var="idioma">
            <li value="NIL"><set var="solucion">No, no esta en <star index="3"/> </set></li>
            <li><value><star index="3"/></value>
            <set var="solucion">Si, <star index="2"/> esta en <star index="3"/></set>
            </li>
            <li>
                <set var="lista"><rest><get var="lista"/></rest></set>
                <set var="idioma"><first><get var="lista"/></first></set>
              <loop/>
            </li>
        </condition>
      </think>
      <get var="solucion"/>
  </template>
</category>
```

de manera que si pregunto “narcos esta en ingles”, responderá “Si, narcos esta en ingles” y si la pregunta es “narcos esta en frances”, dirá “No, no esta en frances”.

Con este tutorial de inicio de la práctica se ha querido ilustrar el proceso que el estudiante debería seguir para confeccionar la misma. Planteándose en primer lugar la forma de responder a preguntas simples, para luego abordar conversaciones más elaboradas como la que se muestra a modo de ejemplo al principio de este tutorial para la inclusión de una nueva serie en una de las plataformas.