Sistemas Inteligentes: Práctica 2

RESUMEN

Este documento detalla la estructura y comportamiento de un agente conversacional capaz de responder a preguntas sobre la Escuela Superior de Ingeniería Informática (ESEI, de aquí en adelante), así como realizar servicios relacionados con la misma o la modificación en caliente de archivos.
El desarrollo de la aplicación se ha realizado empleando el modelo de lenguaje AIML 2.0, usando como motor el lenguaje de programación Java mediante la plataforma derivada Jason. La aplicación es capaz de responder correctamente a las preguntas habituales, variaciones de éstas, aprender de manera supervisada y entablar conversaciones con bastante fluidez. También gestiona servicios en nombre del usuario, según los permisos de los que disponga el mismo.

INTRODUCCIÓN

Los chatbots -o agentes conversacionales-, son una interfaz de interacción entre el usuario y una máquina [1]. La aparición de éstos se remonta al año 1964 [2], con la creación de *Elizabeth* por parte Joseph Weizenbaum, un investigador del MIT. Años más tarde, en 1995, se desarrolla una versión mejorada del mismo llamada *Alice* [2]. La principal diferencia respecto a su predecesora era su habilidad para procesar el lenguaje natural por medio de un entorno web. *Alice* utilizaba patrones para conversar con el usuario [3] y guardaba estos datos en objetos llamados AIML. El resultado del ejercicio de esta práctica tendrá como consecuencia un agente conversacional similar a *Alice*, dado que el modelo de lenguaje AIML usado durante la programación ha sido derivado de los objetos de datos mencionados anteriormente.

APLICACIÓN

La aplicación desarrollada consta de dos infraestructuras bien diferenciadas. Por un lado, tenemos la programación basada en AIML 2.0, que se encarga de generar respuestas en base a una serie de datos de entrada (en este caso preguntas) y; por otro lado dos agentes diseñados e implementados usando el lenguaje Jason, encargados de simular conversaciones o filtrar respuestas con el fin de ejecutar acciones y servicios.

AIML

Dentro de la carpeta de nuestro bot, llamada proyecto, existen numerosos archivos .aiml según la temática que englobe a las diferentes expresiones definidas en cada uno de ellos.

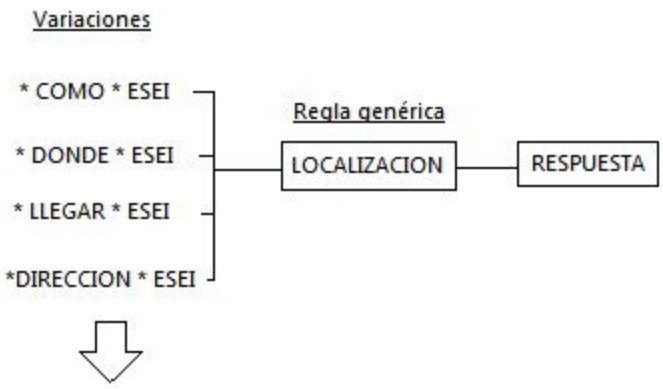
Hemos numerado los archivos usando un prefijo de dos dígitos. El contenido de los archivos es fácilmente distinguible según el sector al que pertenece. Esto facilita la programación, revisión y corrección de errores.

Los rangos según la temática son:

Inicio	Final	Descripción
01	18	Conocimiento tq, ntq (Práctica 1)
20	20	Conocimiento aq
80	88	Servicios (Práctica 2)
97	97	Aprendizaje supervisado
98	99	Archivos genéricos

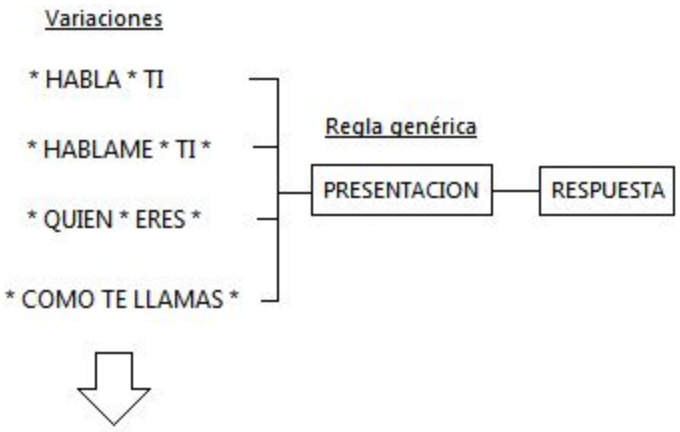
Las respuestas se dan mediante el uso expresiones reducidas gracias al uso de palabras clave y comodines. Además, distintas variaciones suelen dar lugar a la misma respuesta para lograr robustez a la hora de conversar.

Ejemplo de conversación referente a la ESEI:



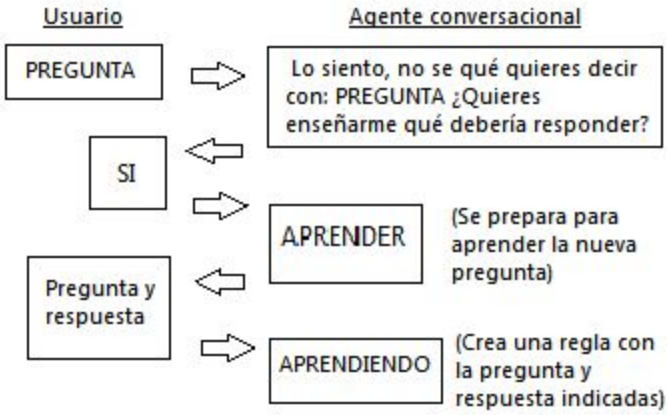
El agente conversacional también es capaz de resolver algunas cuestiones ajenas a la ESEI pero unidas a su personalidad. Para lograrlo, la construcción de las expresiones es similar a la detallada en el punto anterior.

Ejemplo de conversación referente al bot:



Por otro lado, en caso de que la conversación desemboque en terminología o expresiones desconocidas para el agente, mediante la implementación de reglas basadas en contextos y la etiqueta *learn*, es posible ampliar paulatinamente la base de conocimientos del mismo.

Un ejemplo de esto sería una conversación donde le preguntemos algo totalmente nuevo al agente. Una pregunta del estilo *"¿Cuál es mi color favorito?"* generará una disculpa por parte de la inteligencia acompañada de un pregunta cerrada sobre si el usuario desea enseñarle la respuesta a dicha pregunta desconocida.



El final de este diálogo tendrá como consecuencia la adición de nuevas expresiones a la base de conocimiento.

Por último, en el AIML se encuentran las reglas que lanzan mensajes XML como respuesta, diseñados para ser parseados por un agente externo (en este caso *master*, aunque ya hablaremos de ello más adelante). Estas reglas no tienen intención de llegar al usuario final, por lo que las respuestas no constan de ningún mensaje personal o ligado a la conversación.

Un ejemplo de este tipo de contestación es:

```
<mail>
  <to><bot name="email"/></to>
  <subject>Hola <get name="login"/></subject>
  <msg>Funciona</msg>
</mail>
```

Para proteger los servicios lanzados por estas respuestas en crudo, el agente conversacional ha sido dotado con un sistema de *login* permitiendo ejecutar las acciones sólamente a usuarios con privilegios. Estos usuarios están designados en una lista blanca llamada *autorizados*.
El método de logueo es sencillo. Sólamente hay que especificar tu nombre y el bot se encargará de manejar esa información acorde a tu status.

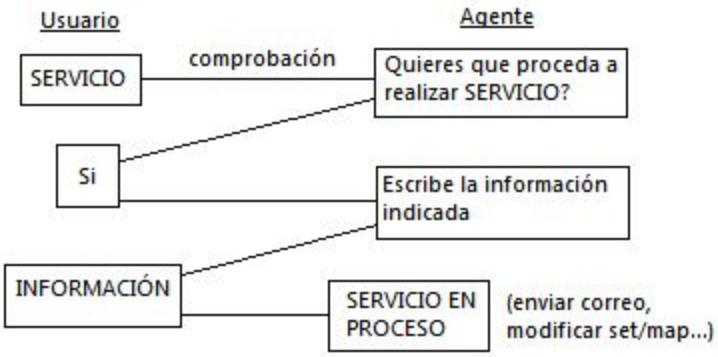
ASL

La otra parte fundamental del programa está sostenida sobre dos agentes ASL llamados *student* y *master*. Ambos hacen uso de las funciones proporcionadas por el entorno *Chatter* para comunicarse con el bot AIML.

El primero de los agentes nombrados ejecuta el papel de un estudiante que conversa con el bot.
En nuestro caso hemos definido una conversación basada en 4 puntos:

- Conversación inicial: Ambos interlocutores se presentan y hablan irrelevancias.
- Preguntas nt/ntq: El estudiante realiza aleatoriamente preguntas (6, por defecto) del tipo tq y otras tantas del tipo ntq. La respuesta dada es la misma pero la pregunta es diferente dado que se busca robustez ante una reformulación del mismo tópico.
- Preguntas aq: El estudiante realiza aleatoriamente preguntas (6, por defecto) del tipo aq.
- Servicios: El estudiante exige al bot que le ayude con todos los servicios que éste dispone.

Para la ejecución de servicios, es necesario expresar el deseo de realizar una de las mejoras detalladas e incorporadas en esta práctica, como cambiarse de grupo reducido o darse de alta en una asignatura. Como ya se ha especificado anteriormente, el estudiante debe estar logueado para la satisfactoria ejecución de los mismos.



Un flujo estándar de las interacciones para llevar a cabo el uso de una de estas acciones podría definirse como:

En ciertos casos, es posible que los correos relacionados con el email tarden demasiado en ejecutarse satisfactoriamente. Es por ello que se han declarado 10 segundos de espera base cada vez que es requerido.
El estudiante recibe las preguntas por una única regla condicionada por la ausencia de etiquetas XML y que se elimina una vez procesada. Dicha regla permite manejar a la vez tanto respuestas directas del bot como reformulaciones por parte del *master*.

El otro agente nombrado, *master*, tiene una función de proxy. Consta de una regla *answer* que verifica la existencia de etiquetas XML en la respuesta proporcionada por el bot. Si es así, parsea el contenido y ejecuta los servicios acordes necesarios.

Para mayor riqueza del proyecto, se han declarado 3 acciones que puede realizar el agente *master*:

- mailing: Envía emails mediante la cuenta asociada "spam.kikefontanlorenzo@gmail.com".
- addset: Añade la información requerida a un archivo perteneciente a la carpeta *sets*. Resulta útil en servicios como la inscripción en materias.
- addmap: Añade la información requerida a un archivo perteneciente a la carpeta *maps.* Resulta útil en servicios de asignación, como en aquellos relacionados con el TFG.

Una vez parseado y ejecutado el servicio, el agente suministra, mediante el comando .send, la respuesta apropiada al agente *student*.

> Recordatorio: Para ejecutar las funciones mailing, la cuenta asociada debe fijarse con baja protección.

CONCLUSIÓN

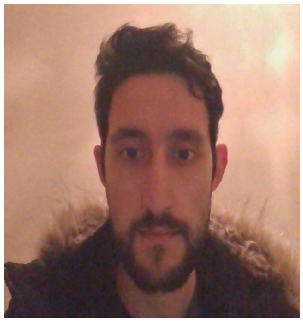
Como conclusiones técnicas, ha sido muy útil el diseño e implementación de agentes ASL que permitieran la automatización y extensión del bot AIML.
ASL es mucho más potente de lo que en un principio pueda llegar a parecer, pero si además lo unimos con APIs de Java, se vuelve un entorno y un lenguaje muy enriquecido.

Como conclusiones personales, me ha gustado usar Jason. He conseguido aprenderlo rápido y se me ha hecho más cómodo que AIML. Por otro lado, esto también aporta un cambio en la monotonía de las prácticas, dado que implica aprender un lenguaje nuevo.
Aún así, el desarrollo de esta práctica ha sido completamente a contrarreloj debido a la sobrecarga a la que estamos expuestos de trabajos y examenes. Esto, unido a la situación actual, las constantes caídas de FaiTIC impidiendo el acceso a los Foros o al Material; o los recientes e intermitentes fallos de servicio de R en toda Galicia, han hecho que realizar este proyecto fuese toda una odisea.

BIBLIOGRAFÍA

[1] Hípola, P., & Vargas-Quesada, B. (1999). Agentes inteligentes: definición y tipología. *Los agentes de información* . Retrieved March 8th, http://www.elprofesionaldelainformacion.com/contenidos/1999/abril/agentes_inteligentes_definicion_y_tipologia_los_agentes_de_informacion.html
[2] Cerdas Méndez, D. (2017). Evolución de los chatbots. *Planeta Chatbot .* Retrieved March 8th, https://planetachatbot.com/evoluci%C3%B3n-de-los-chatbots-48ff7d670201
[3] Abu Shawar, B., & Atwell, E. (2007). Chatbots: are they really useful? *LDV-Forum: Zeitschrift für Computerlinguistik und Sprachtechnologie* , *22* (1), 29–49. Retrieved March 8th, https://doi.org/10.1.1.106.1099

CURRÍCULUM VITAE



Fontán Lorenzo, Diego Enrique
dflorenzo17@esei.uvigo.es

Profesor en el Parque Tecnológico de Galicia, Divulgador Científico, Pentester Red Team y fundador de Hackerspace-La Molinera.

Estudiante de Ingeniería Informática, Closed Beta Developer Amazon Alexa, Closed Beta Developer Facebook SparkAR.

Pombo Graña, Martín
mpgrana17@esei.uvigo.es

Técnico superior en Desarrollo de Aplicaciones Multiplataforma.

Actualmente de estudiante de la Universidad de Vigo para la titulación de Ingeniería Informática. Nivel de inglés B2.