

Práctica 2

Agentes Conversacionales

RESUMEN

En este documento se explica el funcionamiento del agente conversacional, basado en AIML, que es capaz de responder a las dudas más frecuentes que se preguntan los nuevos alumnos en las jornadas de acogida que organiza la Universidad. También explicaremos cómo procesa el lenguaje natural nuestro bot, segmentando las frases palabra por palabra, buscando las palabras clave que nos permiten mostrar la respuesta más acertada.

Además, es capaz de realizar solicitudes de servicios, ampliar su base de conocimiento aprendiendo de las interacciones con el usuario, modificar ficheros, entre otros servicios. Esto lo consigue haciendo uso de Json que filtra las respuestas, realiza los servicios en su nombre y modifica los ficheros persistentes.

INTRODUCCIÓN

Vivimos en el mundo de la información y la queremos rápido. No es casualidad que el uso de Agentes Conversacionales se haya visto incrementado notablemente. Lenguajes como aiml nos permiten automatizar trabajos como atención al cliente, guías, etc. de una forma sencilla pero eficiente. Este tipo de lenguajes reconocen el lenguaje natural, lo procesan y realizan acciones en relación a lo que se le pregunta.

Se ha realizado mucha investigación para combinar y crear tecnologías para simular que se interactúa con un humano y no con una máquina. Estos lenguajes nos permiten crear chatterbots capaces de reconocer y procesar el lenguaje natural, y ofrecer respuestas coherentes. También existen otras tecnologías como el reconocimiento de voz, generación de lenguaje natural o síntesis de voz que, aunque su uso es muy extendido.

Estos lenguajes no solo nos permiten realizar preguntas y que nos responda de manera rápida y coherente, si no que, con la ayuda de lenguajes externos, es capaz de realizar servicios. Para la creación de este chatterbot, hemos utilizado Json y AIML, ya que juntos nos proporcionan las herramientas necesarias para poder realizar el chatterbot con las funcionalidades que buscamos.

DISEÑO DEL PROYECTO

En un primer momento, planteamos el chatterbot para que pudiera responder a preguntas, pero tenían que seguir un patrón específico para encontrar coincidencias. También surgía el problema de que una palabra apareciera en varias reglas, impidiéndonos reconocer a qué regla se refiere en cada caso.

Estos problemas nos llevaron a plantearnos una forma mucho más escalable, recorriendo la frase palabra por palabra en busca de palabras claves que nos permita dar una respuesta adecuada, aunque se pregunte de alguna forma que no se haya contemplado.

Nuestro proyecto tiene dos partes bien diferenciadas, una parte contesta a preguntas que podrían tener los alumnos de la Universidad de Vigo y otra parte que realiza servicios en nombre del usuario tales como solicitar un cambio de grupo, darse de alta en una materia, etc. Comenzaremos con todas las reglas que nos permiten contestar a las preguntas y luego veremos cómo gestionamos los servicios.

SEGMENTACIÓN DE LA PREGUNTA

Para dar respuesta a las preguntas, creamos una regla que la recorre palabra a palabra buscando palabras clave. Las palabras clave que busca se dividen en varios tipos (servicios, pregunta, tema, lugar...) y esta información nos permite dar la respuesta indicada para cada situación. Por ejemplo, en la pregunta: *¿Qué es el PAT?* reconoce las palabras clave *qué* y *PAT*, que se almacenan en las variables globales *pregunta* y *tema* respectivamente.

Para obtener las palabras clave seguimos un proceso iterativo. En cada iteración encontramos una palabra clave, la guardamos en la variable correspondiente y volvemos a buscar más palabras clave hasta que no encuentre ninguna más. Por lo tanto, para cada palabra clave es necesario crear una regla y todas tienen la misma estructura:

1. Un patrón con la palabra o conjunto de palabras clave (para las que usamos los *sets*)
2. Si la palabra a buscar coincide con el patrón, inicializa una variable global con la palabra correspondiente.
3. Volvemos a buscar devolviendo la pregunta sin la palabra clave encontrada.

De esta forma, es capaz de encontrar las palabras clave en la frase para identificar la respuesta que más se ajusta sin importar el orden en el que estén las palabras clave. Por ejemplo, nuestro bot daría la misma respuesta a la pregunta *¿qué es el pat?* y *¿me hablaron del pat, qué es?*, ya que los dos contienen las palabras clave *qué* y *pat*, siendo la información necesaria para saber qué responder.

VARIABLES GLOBALES

Como estábamos comentando, utilizamos las variables globales para guardar las palabras claves a lo largo de las iteraciones que se realizan en la segmentación de la pregunta. Estas palabras clave son las que nos proporcionan la información necesaria para poder mostrar la respuesta más adecuada y las agrupamos en 7 variables globales distintas:

- **Tema:** Almacena el tema principal de la pregunta, en ella nos basamos para discriminar a que se refiere la frase. Por ejemplo: *PAT*, *TFG*, *delegación*...
- **Pregunta:** Almacena la palabra clave que se ha utilizado para realizar la pregunta. Por ejemplo: *Qué*, *cuál*, *cuánto*...

- **Lugares:** Almacena el lugar al que se hace referencia. Por ejemplo: secretaria, ESEI, Universidade de Vigo...
- **TemaEspecífico y VerboEspecífico:** En algunos casos, la variable global tema no es suficiente para determinar a qué se refiere una frase, por lo que necesitamos encontrar otra palabra clave más específica en relación al tema para distinguir una pregunta de otra. Por ejemplo: Las preguntas *¿qué es un tfg?* y *¿qué diferencia hay entre un tfg y un tfm?* serían idénticas para el bot sin esta variable global. Si almacenamos la palabra *diferencia* como tema específico podríamos dar una respuesta distinta para cada pregunta.
- **Personal:** Determina si la frase es una pregunta técnica o es una pregunta personal hacia los creadores del bot. Actúa como un booleano ya que solo toma dos valores. Por ejemplo, para distinguir entre *¿qué tfg quieres hacer?* y *¿qué es un tfg?*, necesitamos la variable Personal que las diferencie usando el verbo *quieres* (que está en segunda persona). Las otras dos palabras clave (*qué* y *tfg*) no sirven para diferenciarlas porque son comunes a ambas preguntas.
- **Servicio:** Almacena qué servicio se quiere realizar. Esta variable se complementa siempre con un topic para poder tener reglas específicas para cada servicio. Y conseguir así tratar de forma distinta las preguntas normales de los servicios. Más adelante explicaremos con más detalle con se gestionan los servicios.

Este sistema de variables globales dio lugar a varios problemas que hubo que resolver. En primer lugar, la variable Personal nos limitaba mucho al utilizar los verbos en segunda persona como palabras clave. Los verbos, además de indicarnos si se trata de una pregunta personal o no, tienen un significado que no aprovechamos. Para resolverlo, hacemos uso de los *maps* y en vez de eliminar el verbo en segunda persona lo sustituimos por el verbo en infinitivo y volvemos a buscar más palabras clave.

Otro problema que tuvimos estaba relacionado con la variable Pregunta. Existen frases en las que se utilizan más de una palabra perteneciente a esta categoría. Por ejemplo, *¿cómo quieres que te llame?* contiene las palabras *cómo* y *que*, ambas serían identificadas como palabras clave y, puesto que solo hay una variable Pregunta, la última en ser encontrada sobrescribirá a la primera.

En nuestro ejemplo, el programa podría considerar incorrectamente que *que* es una palabra clave cuando debería ser *cómo*. Para solucionar esto, pensamos en forzar que la palabra clave correspondiente a la variable Pregunta estuviese al inicio de la frase para ser considerada como tal. Pero el usuario podría preguntar de manera indirecta (*¿en qué consiste el PAT?*, por ejemplo). Por eso después de buscar todas las palabras clave, si no sabe qué respuesta dar, elimina la primera palabra de la pregunta y vuelve a buscar por si encuentra una partícula interrogativa. En nuestro ejemplo eliminaría la palabra *en*.

ELECCIÓN DE LA RESPUESTA

Para la elección de la respuesta correspondiente, hemos creado una función que utiliza el valor almacenado en la variable Tema. Dentro de cada uno de los valores que puede tomar dicha variable, entran en juego el resto de variables que nombramos en el apartado anterior. Por ejemplo: en la frase *¿que es el pat?*. Como ya dijimos antes, las variables globales son

Tema=PAT y Pregunta=que, entonces la respuesta a esa pregunta estará almacenada en el condicional Tema=PAT y dentro de ese condicional, el condicional Pregunta=que.

Para hacer nuestro proyecto más robusto, creamos un sistema para que, en caso de encontrar un tema pero no una respuesta específica, proponga la pregunta que más coincidencia tiene con la que ha formulado. Por ejemplo, si un usuario pregunta *¿Cuál es el PAT?*, no se encontraría específicamente esa pregunta, pero el bot si sabrá que se refiere al tema pat, por lo que le dirá *Tu pregunta es: que es el pat?*. Si contesta que sí, muestra la respuesta a *¿que es el pat?* y aprende que si le vuelven a preguntar *¿cuál es el pat?* debe responder como si hubieran preguntado *¿qué es el pat?*. En el caso de no encontrar ninguna coincidencia, devuelve el mensaje *Mis respuestas son limitadas*, la respuesta por defecto siempre que no encuentra una respuesta adecuada.

SERVICIOS

Con AIML conseguimos el chatterbot que hemos comentado a lo largo del documento, pero para poder realizar determinados servicios y modificaciones de ficheros persistentes (básicamente maps y sets), utilizamos AgentSpeak. Para ello, hemos integrado el bot en un proyecto de Jason con dos agentes en el lenguaje AgentSpeak, uno que formula preguntas (Student) y otro que realiza los servicios (master). Los dos reciben los mensajes del bot, pero son ellos los encargados de filtrar las respuestas para saber a quién va dirigido el mensaje, student realiza preguntas al bot y captura la respuesta del bot ,y master es el encargado de realizar los servicios y modificaciones.

Primero procederemos a explicar cómo gestiona el bot los servicio y por último veremos cómo funcionan los agentes Master y Student.

CONTROL DE USUARIOS

Decidimos crear un controlador de inicio de sesión para que los usuarios no tengan que escribir su nombre y dni cada vez que quieran solicitar un servicio. Cada usuario se identifica con su nombre y dni, manteniendo dichos datos almacenados hasta que decida cerrar la sesión. Además, existe un par de valores para que el administrador del sistema se pueda loguear como tal y poder añadir datos, como por ejemplo añadir TFGs propuestos, sin tener que modificar el código. Estamos hablando de modificar ficheros persistentes por lo que realmente es el Agente Master quien realiza la modificación.

ELECCIÓN DE SERVICIOS

Para la elección de servicios, planteamos seguir la estructura que teníamos anteriormente, pero generaba conflicto en muchas reglas. Se nos ocurrió crear un topic para cada servicio y poder hacer una conversación guiada para mandar el correo correspondiente. Este método tenía demasiado código duplicado, por lo que lo dividimos en varios topics, uno para elegir el servicio, otro para confirmar que los datos proporcionados están correctos antes de enviar el correo, etc. Así, podemos reutilizar código entre los servicios, sin entrar en conflicto en las reglas ya que están en diferentes topics. La estructura que sigue la elección de servicios es:

Primero el chatterbot reconoce que está realizando la solicitud de un servicio, entra en el topic específico que determina qué servicio quiere realizar (cambio de grupo, alta en una asignatura..), una vez sabemos que servicio es, accede al topic servicio y, si es la primera

vez que hace una solicitud, le pide su nombre y dni, que actuará como inicio de sesión, evitando que tenga que escribir estos datos cada vez que quiera realizar un servicio en esa sesión.

Una vez introduce el nombre y el dni, ya se accede al topic servicio específico y hace las preguntas correspondientes a cada servicio. Cada servicio tiene sus comprobaciones para que el correo tenga sentido, por ejemplo, antes de aceptar la defensa de un TFG, comprueba que ese TFG exista y que concuerde con la ID correspondiente. Una vez obtiene toda la información necesaria, envía un correo solicitando el servicio en su nombre y sale del topic. En el caso de que quisiera realizar otro servicio, el procedimiento sería el mismo pero ya no tendría que identificarse con su nombre y dni. Para poder verlo mejor, veamos una posible conversación con el bot:

human: *Quiero solicitar cambio de grupo reducido // El usuario solicita realizar un servicio*

Busca palabras clave y, en este caso, encuentra la palabra solicitar que nos indica que se está realizando un servicio, por lo que inicializa la variable topic a ELECCIONSERVICIO. A continuación encuentra el patrón grupo reducido dentro de las reglas propias del topic. Aquí se cambia el topic a servicio y la variable servicio a grupo reducido. Como el usuario no tiene sesión iniciada, le pide su nombre y dni.

bot: *En primer lugar necesito que te identifiques. Cual es tu dni?*

human: *Mi dni es 11111111A*

bot: *Cual es tu nombre?*

human: *Mi nombre es Antonio Morales*

Se añade al set dni.txt dicho dni y añade el par dni:nombre al map usuarios. A partir de ahora recordará el nombre del usuario y simplemente con poner el DNI sabrá quien es.

bot: *Hola Antonio. Para solicitar el cambio de un grupo reducido, necesito que me digas en qué materia quieres cambiar de grupo. P.E: En [Sistemas inteligentes].*

human: *En sistemas inteligentes*

Comprueba que "Sistemas inteligentes" esté registrado como una asignatura en el sistema y si lo está, la almacena.

bot: *Entiendo, en la asignatura de sistemas inteligentes. En qué grupo estás actualmente?[P.E:En el grupo 6]*

human: *En el grupo 2*

Comprueba que sea un número y almacena el grupo en el que está actualmente.

bot: *De acuerdo, estás en el grupo 2. A cual te quieres cambiar?[P.E: Al grupo 3]*

human: *Al grupo 1*

Comprueba que sea un número y que no sea el mismo grupo que en el que está actualmente.

bot: *El correo que se enviará contiene el siguiente mensaje: Buenos días, soy Antonio, me gustaría cambiarme del grupo reducido 2 al grupo reducido 1 de la asignatura sistemas inteligentes. Es correcto?*

Pide confirmación, si confirma que está todo correcto realiza el servicio, si no, elimina los valores de las variables y cancela el proceso.

human: *Sí*

Se envía un mensaje con la etiqueta mail que el agente Master interpreta y le permite enviar el correo. Una vez lo envía, se olvida el valor de la variable topic para que en la siguiente interacción del usuario no pueda acceder a las reglas específicas del servicio.

De este modo funcionan todos los servicios, siguiendo una estructura guiada, similar las unas a las otras, tan solo cambiando los datos necesarios para solicitar el servicio.

AGENTE STUDENT

El agente student actúa como si fuese un humano. Tiene como objetivo principal interactuar con el bot, tanto realizándole preguntas como pidiéndole la realización de servicios. Por eso, su plan inicial es la realización de una serie de preguntas aleatorias al bot de entre las que tiene almacenadas en su base de conocimiento. La dinámica general es que Student realiza preguntas sin tener en cuenta la respuesta del bot (tan solo mostrando la respuesta recibida) pero hay situaciones en las que Student debe reaccionar a la respuesta del bot.

Por un lado están las respuestas del estilo *Tu pregunta es: que es el pat?*. En este caso Student identifica si realmente se refería a eso y en caso afirmativo responde con un si. De esta forma el bot aprende y da la respuesta adecuada.

Por otro lado están los servicios que requieren de una interacción constante con el bot. Para resolver esto se han planteado 8 conversaciones predefinidas a sabiendas de lo que el bot va a responder. De entre las 8 conversaciones, Student elige 3 de forma aleatoria. Además, para evitar tener que identificarse para cada servicio y puesto que el bot solo nos exige identificarnos para el primer servicio, Student reacciona y se identifica sólo cuando el bot se lo pide. Esto último, Student lo hace siempre que el bot le pida que se identifique y es independiente de las 8 conversaciones predefinidas.

En otro orden de cosas, había un problema a la hora de gestionar las respuestas del bot repetidas. Cuando a dos preguntas consecutivas el bot daba la misma respuesta, la base de conocimiento de Student no cambiaba por lo que no reaccionaba, como si el bot no hubiera respondido. Para arreglarlo, decidimos borrar de la base de conocimiento del bot las preguntas justo cuando las recibe.

Por último, comentar que Student ignora las respuestas del bot que se corresponden con servicios, ya que estas son atendidas primero por Master y será este el que finalmente le dé una respuesta a Student.

AGENTE MASTER

El agente Master tiene como objetivo realizar los servicios que recibe del bot y enviarle al agente Student el resultado de dichas operaciones haciéndose pasar por el bot. Para ello, al contrario que Student, ignora todos los mensajes que no se corresponden con servicios y reacciona en el resto de casos.

Cuando el bot quiere realizar un servicio lo que hace es enviar la respuesta con la información necesaria para su realización encapsulado con etiquetas propias de cada servicio que el Master filtra y obtiene el valor correspondiente.

Por ejemplo para el servicio de enviar un correo la estructura sería utilizando la etiqueta *mail* para identificar que es un servicio que debe realizar el Master. Dentro de esta etiqueta, encontramos el asunto con la etiqueta *subject*, el correo electrónico a quien se le envía el servicio con la etiqueta *to* y el cuerpo del mensaje con la solicitud del servicio con la etiqueta *msg*.

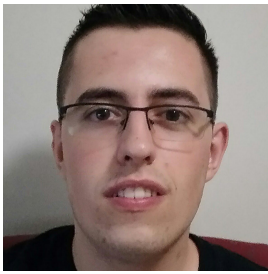
Para determinar que una respuesta es un servicio, al igual que hace Student, Master busca en la respuesta alguna de las etiquetas correspondientes a estos. De esta forma no solo sabe que tiene que realizar un servicio sino que también sabe cuál debe realizar. Una vez sabe cuál debe realizar sigue una serie de reglas específicas de ese servicio que le permite extraer toda información para que finalmente puede llevar a cabo el servicio. Finalmente, notifica a Student de que el servicio se ha realizado correctamente.

CONCLUSIÓN

En este proyecto, hemos utilizado el lenguaje AIML y Agentspeak para poder comprender el lenguaje natural, estos lenguajes son completamente distintos a los que estamos acostumbrados, por lo que intentamos adaptar el código de tal forma que fuera lo más sencillo posible, basándonos en funcionalidades que encontramos en otros lenguajes, como recorrer una lista de caracteres, que estos lenguajes no implementan de forma nativa.

Este bot puede ser implementado por la Universidad de Vigo para que los alumnos puedan encontrar respuesta a las dudas más comunes que suelen surgir a lo largo del grado, ya que responde desde donde se encuentra la cafetería hasta solicitar la defensa de un TFG. Además, al ser capaz de añadir información mediante la propia interacción con el bot, puede ser ampliado de forma sencilla y persistente.

En conclusión, nuestro bot es un chatterbot eficiente a la par que robusto y escalable, que permite responder preguntas, aprender de las interacciones de los usuarios y puede realizar servicios en nombre del usuario conectado.



Sergio García Nuñez

sergiogarnu@gmail.com

Estudios:

Bachillerato de ciencias y tecnología.

Estudiante de Grado en Ingeniería Informática.

Idiomas:

Inglés nivel medio

Gallego nivel nativo



Luciano Martín González Van Gent

lggent17@esei.uvigo.es

Estudios:

Bachillerato de ciencias y tecnología.

Estudiante de Grado en Ingeniería Informática.

Experiencia laboral:

Camarero los fines de semana.

Idiomas:

Inglés nivel medio

Gallego nivel nativo



Rubén Vilas Martínez

rvmartinez@esei.uvigo.es

Estudios:

Bachillerato de ciencias y tecnología.

Estudiante de Grado en Ingeniería Informática.

Experiencia laboral:

Camarero los fines de semana.

Becario de beca de formación en 2017-2018

Idiomas:

Francés nivel básico

Inglés nivel avanzado

Gallego nivel nativo

