Inteligencia artificial para la realización de procesos comunes mediante Watson, NodeJS y Blue Prism

Jose María Bello de Haro

Resumen– Bluemix es una plataforma que ofrece multitud de servicios: Bases de Datos (BBDD), compilación, despliegue de aplicaciones... Y de entre todos estos servicios, aquellos que usan la Inteligencia Artificial (IA) de Watson y ML son los que se exponen. Uno de ellos, Watson Conversation, se implementa para poder crear un asistente virtual para la UAB, el cual podrá interactuar y responder al usuario que solicite alguna acción a través de un chat programado mediante NodeJS. Además, otras acciones serán llevadas a cabo mediante Blue Prism, como el envío de correos o el registro de usuarios. Watson hace uso de 'Machine Learning' (ML) para poder analizar, aprender y responder al usuario como si de una persona real se tratase. Esto se consigue mediante el entrenamiento de modelos personalizados.

Palabras clave– Watson, Inteligencia Artificial, Bluemix, Blue Prism, NodeJS, Base de Datos, Machine Learning, Conversation, Intención, Entidad, Nodo, Aprendizaje, Despliegue, SOAP, Restful API, JSON, IBM, IBM Cloud, Entrenamiento, Test, Groundtruth, Exhaustividad, Precisión, Matriz de Confusión, validación cruzada

Abstract– Bluemix is a platform that offers multiple services: Databases, compilation, application deployment... And between all this services, those that use Watson Artificial Intelligence (AI) of Watson and ML are those that are exposed. One of them, Watson Conversation, is implemented to create a virtual assistant for the UAB, which can interact and respond to the user requesting some action through a chat programmed through NodeJS. In addition, other actions will be carried out through Blue Prism, such as mailing or user registration. Watson uses 'Machine Learning' (ML) to analyze, learn and respond to the user as if it were a real person. This is achieved by training custom models.

Keywords– Watson, Artificial Intelligence, Bluemix, Blue Prism, NodeJS, Database, Machine Learning, Conversation, Intent, Entity, Node, Learning, Deployment, SOAP, Restful API, JSON, IBM, IBM Cloud, Training, Test, Groundtruth, Recall, Precision, Confusion Matrix, cross-validation

1 Introducción

ACTUALMENTE está en auge el uso de asistentes virtuales para gestionar multitud de tareas que anteriormente se llevaban a cabo mediante personal de atención 24h o asistentes virtuales gestionados por personas. También la búsqueda de información se hace laberíntica a medida que el sitio web del que se desea obtener información es más grande.

Es por ello que con el presente trabajo se lleva a cabo la creación de un **asistente virtual para la UAB**. El asistente virtual, mediante IA, permite automatizar todas las tareas. Y actualmente no existe rival para las capacidades cognitivas de Watson. Existen opciones que se centran en algunos aspectos concretos, como el análisis del lenguaje natural [1] o la clasificación de imágenes [2]. Pero al final Watson ofrece una alternativa para todo, más o menos potente. Y es que es esa la ventaja: tener bajo un mismo abanico multitud de servicios que se pueden interconectar entre ellos y aprovechar todo el potencial que nos ofrece Watson (y IBM Cloud). Algunas de las empresas que emplean a Watson en

• E-mail de contacto: josemaria.bello@e-campus.uab.cat
• Mención realizada: Computación
• Trabajo tutorizado por: Fernando Luis Vilariño Freire (Ciencias de la Computación)
• Curso 2017/18

sus sistemas son: General Motors [3], la America Cancer Society [4], Wimbledon [5], Rocket Fuel [6], entre multitud de otras empresas importantes [7] [8]. Con esto podemos comprobar lo hondo que está calando el uso de IA para llevar a cabo todo tipo de tareas. Ya sea reconocimiento de voz, reconocimiento de caracteres, análisis de la actitud o el estado de ánimo, análisis de la personalidad a partir de las palabras, aprendizaje a partir de las conversaciones, Watson ofrece multitud de alternativas y todas interconectadas. Y es por eso que el número de empresas que hacen uso de éste crece cada año. Con todo este background y teniendo en cuenta el crecimiento que está teniendo, se ha decidido implementarlo en el actual proyecto. Además de que tanto el soporte online como la documentación es fácil de encontrar y de acceder.

2 Estado del Arte de IBM Watson

Watson consiste en una plataforma de computación cognitiva, lo que significa que el contenido recibido mediante diversas entradas puede ser procesado y analizado, gracias al uso de algoritmos, Inteligencia Artificial y Big Data, para posteriormente dar una respuesta lo más precisa posible. Con ello se espera poder ofrecer un sistema que sea capaz de desenvolverse en situaciones complejas que requieren la emisión de respuestas elaboradas y razonadas. Para conseguir llevar esto a cabo se diseñó la arquitectura DeepQA, que fue motivado por el reto que se propuso un equipo de IBM de participar en el concurso televisivo estadounidense "Jeopardy!" y ganarlo con un sistema informático.

2.1. Arquitectura DeepQA

DeepQA (Deep Question Answering) [9] conforma un sistema que es capaz de responder en tiempo real a preguntas formuladas con lenguaje natural. A grandes rasgos permite generar hipótesis, recopilar pruebas de forma masiva, analizarlas y clasificarlas. Dicho sistema está formado por diversas etapas:

1. **Adquisición de contenido**: se lleva a cabo mediante Big Data, Wikipedia u otros orígenes. Estos se parsean en grupos sintácticos, buscando así detectar sujeto, verbo, objeto... Tras ello se busca generalizarlos en grupos semánticos, buscando la relación entre las diferentes partes. Con todo ello se crea la red semántica.

2. **Análisis de la pregunta**: en este parte se busca entender aquello que se ha preguntado. Para poder sacar todas las posibilidades, se toman diferentes acercamientos para poder abarcar todas las posibles interpretaciones de la pregunta. Y para facilitar el análisis, se divide la pregunta en subpreguntas que tengan más fácil respuesta.

3. **Generación de hipótesis**: a partir de toda la base de datos que se tiene se lleva a cabo una primera búsqueda para poder obtener la máxima cantidad posible de información para elaborar la respuesta. Esto se lleva a cabo mediante el uso de diversas técnicas de búsqueda existentes, como las "triple store queries", motores de búsqueda de texto múltiple o búsqueda de documentos, entre otras. Tras la búsqueda, se extraen respuestas que sean posibles candidatas a responder a la pregunta formulada. Estas respuestas pueden ser títulos en el caso de documentos, texto de pasajes... Es por ello que en esta etapa no se busca precisión si no obtener una base de candidatos. El resto de etapas se irán encargando de depurar la respuesta. Si en esta etapa no se encontrado una respuesta candidata lo más probable es que no se pueda responder a la pregunta formulada.

4. **Puntuación de hipótesis**: una vez las respuestas candidatas han pasado el "soft filter", que consiste una serie de algoritmos que permiten reducir la cantidad de candidatos antes de pasar a la etapa actual (siendo dicha cantidad un parámetro a escoger), han de pasar un proceso de evaluación estricto que permite obtener pruebas adicionales para poder dar soporte a cada respuesta candidata. Esto permite puntuar cada respuesta según las pruebas obtenidas. Algunos métodos de puntuación de respuestas son geoespaciales, temporales, relacionales, popularidad...

5. **Unir resultados**: se identifican las respuestas relacionadas y se unir sus puntuaciones. Tras ello se debe establecer un ranking de las hipótesis y calcular la confianza basado en los resultados de la unión. Utilizan "Machine Learning" sobre un conjunto de preguntas de entrenamiento con respuestas conocidas y entrenando un modelo basado en las puntuaciones.

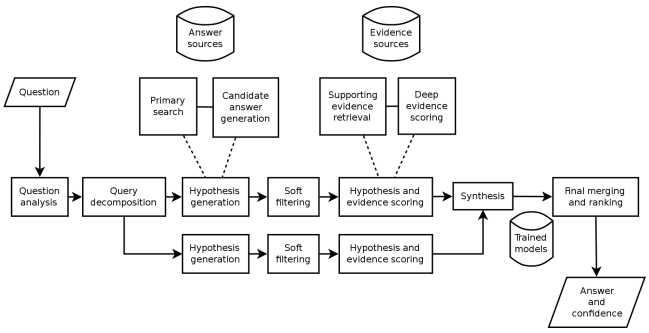

Fig. 1: Arquitectura del DeepQA

2.2. ¿Qué posibilidades ofrece Watson?

En los últimos años las capacidades de Watson han ido aumentando significativamente, abarcando cada vez más ámbitos, ya que el objetivo es que se convierta en un asistente virtual que sea capaz de responder lo más humanamente posible a los usuarios. Actualmente, las posibilidades ofrecidas por Watson van desde el análisis del lenguaje natural hasta el análisis de vídeo, pasando el análisis de audio. Todo ello ha sido necesario debido a que si no hubiese evolucionado y hubiese permanecido únicamente como DeepQA, entonces sería incapaz de analizar y detectar la multitud de características que a continuación se exponen:

2.2.1. Conversation

Es uno de los servicios más utilizados en el ámbito internacional. Y es el que se usa en el presente proyecto. En

pocas palabras, recibe como entrada lenguaje natural y, mediante *Machine Learning* responde, de forma estructurada y coherente, como si fuese una persona.

Dicho servicio utiliza la IA de Watson y engloba en un único servicio *Natural Language Classifier*, *Natural Language Understanding* y *Dialog*, servicios que se exponen en el apartado 2.2.2 y en el apéndice. Conversation hace uso de HTML, Javascript y JSON para poder transmitir y recibir datos, así como configurar funcionalidades específicas directamente en la conversación. La conversación en sí permite identificar intenciones y entidades. Las intenciones hacen referencia a aquello que ha dicho el usuario (quisiera poder descargar...). Las entidades, en cambio, son los valores que nos interesa identificar entre todo lo que ha dicho el usuario (calendario, horario, clase). Por otra parte, está la lógica de la conversación, que dependiendo de lo que diga el usuario, la conversación avanzará por un sitio u otro. Esta lógica consiste en la configuración de una serie de nodos que formarán el árbol completo de posibilidades. Es en estos nodos donde se implementa el código HTML, Javascript y JSON. JSON permitirá pasarle todos los datos necesarios al orquestador, entre los que están las variables de contexto, que son valores obtenidos de la conversación o generados para gestionar el avance en la conversación.

Lo primero que se tendrá que hacer es crear un servicio de Conversation [10] [11] en Bluemix. Éste servicio nos ofrece una interfaz de lenguaje natural que se añade a la aplicación desarrollada. Esto nos dará opción a generar diversas conversaciones en un workspace. Cada conversación tiene un identificador único y usan:

- **Intents**: hace referencia a todo aquello que el usuario puede querer hacer con nuestra aplicación y todo aquello que queremos manejar. Podemos controlar palabras como: comer, dormir, descargar... Es decir, representan la acción a llevar a cabo.
- **Entities**: clarifican la acción a llevar a cabo, es decir, serían los nombres que acompañan al verbo. Por ejemplo: si "descargar" es el verbo, "PDF horarios ingeniería informática" sería el nombre.
- **Dialog**: es todo el árbol a seguir según los intents y entities detectados. Sería la lógica simple a seguir.

Por otro lado, se tendrá que gestionar la compilación y el despliegue de la aplicación en la nube. En este caso, Bluemix ofrece varios servicios que interconecta mediante los módulos "Availability-monitoring-auto" y "Continuous Delivery". El código quedará subido en github, y mediante los módulos mencionados se compilará y se desplegará online.

2.2.2. Otros servicios

En este apartado se describe de forma resumida la función de cada uno de los servicios. En el apéndice puede obtenerse algo más de información sobre ellos.

Discovery

Mediante el uso de análisis de datos y la capacidad cognitiva de Watson, permite llevar a cabo la creación de aplicaciones de tipo cognitivo que permitan obtener información procesable a partir de información desestructurada. Esta información puede ser propia, pública o de terceros.

Knowledge Studio

Se utiliza para crear un modelo de *Machine Learning* que sea capaz de entender los matices del lenguaje, el significado y relaciones específicas.

Language Translator

Permite identificar el idioma en que se ha escrito y traducirlo a otro. A éste se le puede entrenar para que lleve a cabo traducciones específicas.

Natural Language Classifier

Como bien dice su nombre, es un clasificador que permite comprender el lenguaje de textos cortos mediante técnicas de ML, identificando intenciones, clasificando y organizando datos.

Natural Language Understanding

Permite analizar las características sintácticas del texto de entrada: categorías, conceptos, emociones, entidades, palabras clave, metadatos, relaciones, roles semánticos y sentimiento.

Personality Insights

Mediante el análisis lingüístico del texto (tweets, emails, mensajes de texto...) busca obtener las características de la personalidad del autor.

Speech to Text

Para transcribir con precisión la voz humana, aprovecha la inteligencia de la máquina para combinar información sobre la gramática y la estructura del lenguaje con el conocimiento de la composición de la señal de audio.

Text to Speech

En este caso, a partir de un texto Watson busca, mediante su sintetizador de voz, obtener una voz lo más natural posible.

Tone Analyzer

El analizador de tono permite analizar un texto y detectar el tono y las emociones del mismo. Es capaz de extraer dicha información tanto de un documento entero como de una frase. Esto permite, por ejemplo, analizar el tono de un correo antes de enviarlo, para así poder evitar malos entendidos.

Visual Recognition

Mediante algoritmos de "Deep Learning" analiza imágenes y extrae objetos, caras, escenas y otro contenido. La respuesta que da incluye una serie de palabras clave que otorgan información sobre el contenido de la imagen.

2.3. ¿Para qué se usa en el mundo?

Aunque Watson y el DeepQA comenzó con el objetivo de ganar el concurso televisivo "Jeopardy!", en el que querían ser capaces de hacerle analizar en tiempo real el lenguaje natural y poder ofrecer una respuesta lo más precisa posible, se vió que la capacidad de procesamiento del lenguaje natural (NLP) [12] y de razonamiento de Watson podían utilizarse para otros ámbitos, ya que es capaz de extraer, de un gran conjunto de datos, información relevante y útil. Existen una gran cantidad de ejemplos, como:

- Generar una serie de diagnósticos sobre el estado de un paciente a partir de información obtenida de diversos orígenes, como lo son enciclopedias, radiografías, resonancias magnéticas, resultados de analíticas, historiales médicos... Estas implementaciones permiten aumentar la precisión en los diagnósticos, lo que supone una mejora en el cuidado y tratamiento de los pacientes [13]
- Identificar proteínas adicionales de ARN alterado por la ELA [14]
- Análisis visual del cáncer a nivel mundial, partiendo de los datos ofrecidos por la OMS (Organización Mundial de la Salud). Ofrecer información sobre las regiones y los diversos tipos de cáncer, la edad, el género... [15]

Aun así, en este apartado se van a exponer de forma más concreta y precisa dos casos concretos: el uso de la computación cognitiva para aprender programación paralela; y el diseño y análisis de la percepción visual de un niño usando el servicio de análisis multimedia de IBM.

2.3.1. Uso de la computación cognitivia para aprender programación paralela [16]

El objetivo de este caso es ser capaz de usar el sistema cognitivo de Watson para la educación de personas que comienzan a programar en paralelo. Se busca evitar la máxima cantidad de fallos a la hora de programar y usar las directivas de OpenMP (software libre para programar sistemas paralelos). Para ello han desarrollado una herramienta que usa el NLP de Watson para así evitar errores comunes. El usuario que se conecta a la aplicación realiza una pregunta sobre OpenMP, Watson la analiza y ofrece una respuesta en caso de que la haya.

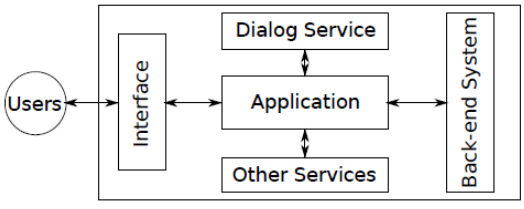

Fig. 2: Arquitectura de la aplicación de ayuda a la programación paralela

Pero para que Watson sea capaz de analizar y elaborar una respuesta, ha de ser entrenado de alguna manera. Y es por ello que utilizan como datos de entrenamiento los errores más comunes que se dan al usar OpenMP, que en este caso alcanzan la cantidad de 32 errores conocidos [17]. Una vez entrenado, se configuran también los valores que van a permitir que el servicio haga uso de una lengua u otra, que se retroalimente... Y con dicha configuración, también se generan otros archivos que incluirán la bienvenida, así como las variantes en lo que a contestaciones se refiere: existe el "input tag", el "output tag", el "default tag", etcétera; los cuales permiten ir configurando el diálogo. Con todos los valores y las etiquetas definidas, y una vez funcionando, se llevó a cabo una encuesta a aquellos usuarios noveles que usaron la aplicación. Con esta encuesta se buscaba conocer cuán útil ha sido la herramienta desarrollada a partir de 4 simples preguntas:

1. ¿Es más útil esta herramienta que el uso de un buscador o un paper a la hora de programar en paralelo? Más del 87,5 % la encontraba útil.
2. ¿Las respuestas dadas fueron precisas? 62,5 % considera que las respuestas fueron precisas.
3. ¿Encuentras útil que sea multilingüe? El 87,5 % vieron con buenos ojos el multilingüismo.
4. ¿Sería útil si la aplicación pudiese devolver *papers*? Un 62,5 % consideró útil una mejora para que devuelva *papers* en lugar de una respuesta concreta.

2.3.2. Diseño y análisis de la percepción visual de un niño usando el servicio de análisis multimedia de IBM [18]

Las principales motivaciones que llevan a utilizar el IMARS (IBM Multimedia Analysis and Retrieval System) son:

- la precisión, gracias al uso de un framework basado en características y que hace uso del "Machine Learning" para clasifica y modelar imágenes y vídeo
- tiempo reducido con respecto a Matlab y otros, y más preciso
- libre acceso de múltiples clientes

IMARS básicamente es un sistema que puede ser usado para indexar de manera automática, clasificar y llevar a cabo búsquedas en grandes colecciones de imágenes digitales y vídeos. Aplica una serie de algoritmos que permiten analizar las características y clasificarlos según su contenido. IMARS es compatible con una amplia gama de clasificadores semánticos pre-entrenados que identifican automáticamente si cada nueva imagen y video pertenece a una o más de las categorías semánticas predefinidas en la taxonomía en función de los descriptores visuales extraídos.

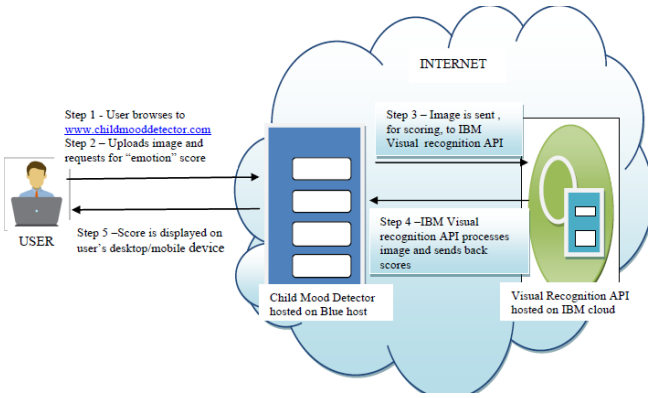

Fig. 3: Arquitectura del VRS

3 ARQUITECTURA DEL PRESENTE PROYECTO

En este apartado se puede contemplar una imagen que muestra arquitectura del sistema que es usada para el presente proyecto y una breve descripción de cada parte (el *conversation* ya ha sido definido anteriormente, ergo se omite en las descripciones).

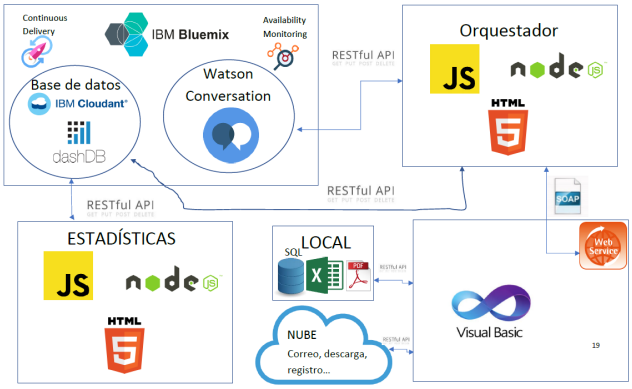

Fig. 4: Arquitectura del sistema propio

3.1. WATSON CONVERSATION

En este apartado se expondrá de forma concreta qué se ha empleado para desarrollar una primera conversación base que permita llevar a cabo pruebas con Watson. Como se ha mostrado en el apartado 2.2.1, consta de Intents, Entities y Dialog. Para poder trabajar con la conversación, harán falta tanto las credenciales como la identificación de la conversación, para poderlas incluir en el apartado de servidor, en NodeJS.

```
{
  "url": "https://gateway-fra.watsonplatform.net/conversation/api",
  "username": "32032295-dc7a-468d-9fb4-74a53ea94767",
  "password": "ulUYPwPh3LF6"
}
```

Fig. 5: Credenciales del conversation del asistente de la UAB

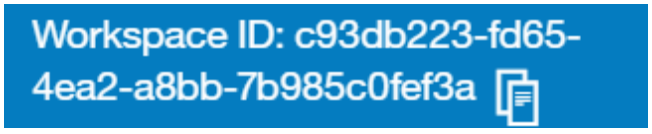

Fig. 6: Identificación de la conversación

3.1.1. Intents

En esta primera conversación se ha tenido en cuenta intenciones del usuario como: descargar o solicitar calendarios, pedir la disponibilidad de algún centro, obtener el correo, el teléfono o la información general sobre algún académico, descargar los horarios de alguna asignatura. Es por ello que se han definido diversos intents con las posibles estructuras gramaticales que pueden usar los usuarios:

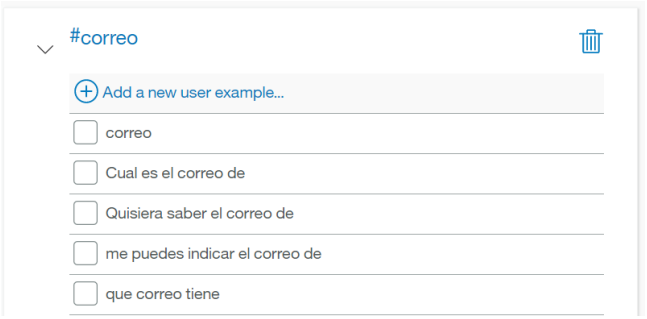

Fig. 7: *Intent* correo con las estructuras gramaticales

Será a partir de estas estructuras gramaticales y del porcentaje de similitud a cada una de ellas como se decida qué ha querido decir el usuario exactamente.

3.1.2. Entities

Basándose en los intents, y, por lo tanto, en aquello que el usuario puede solicitar, se han creado una serie de entities que permiten guardar actualmente: páginas web, información sobre horarios, tipos de calendarios, disponibilidad, carreras de diversos tipos, hora, zonas del campus, facultades, menciones y cursos.

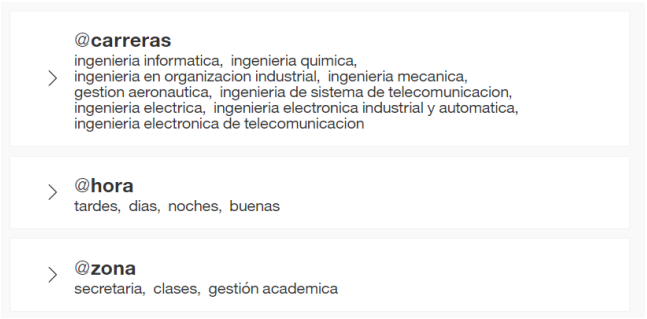

Fig. 8: Entities *carreras*, *hora* y *zona* con los posibles valores

3.1.3. Dialog

En este apartado se han definido las respuestas a darle al usuario, las condiciones a cumplir según lo que pida y la navegación entre los nodos. Éstas son: bienvenida y saludos, disponibilidad, webs, correo, teléfonos y pdfs.

Fig. 9: Nodo *Horario* que comprueba si existen las entities *carreras*, *curso* y *mención*. Guarda los valores en las variables con el mismo nombre.

3.2. CLOUDANT NOSQL DB

Este servicio, descrito más arriba, te ofrece la posibilidad de conectarte desde NodeJS mediante unas credenciales que se generan automáticamente tras la creación:

Fig. 10: Credenciales de Cloudant DB en Bluemix para la conexión

La gestión de la base de datos se lleva directamente desde la propia interfaz gráfica que Bluemix te proporciona. Debido al plan gratuito o limitado, la cantidad de querys, bases de datos a crear, etc. Está bastante limitado. En este caso únicamente se ha creado una base de datos "asistenteuab" en la que se van a guardar los datos que nos interesen del usuario: nombre, id de la conversación, día, hora, valores de las variables de contexto...

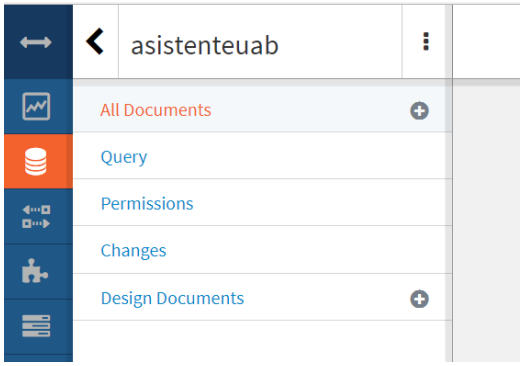

Fig. 11: Interfaz gráfica de gestión de la base de datos Cloudant DB

3.3. ORQUESTADOR

El orquestador o nexo consta de dos partes: la parte del cliente, que implementa Javascript puro; y la parte del servidor, que implementa NodeJS. A continuación, se exponen ambas partes con sus módulos y funciones principales.

3.3.1. Cliente

Es la parte que conforma el front-end e incluye la interfaz gráfica (HTML y CSS) y la lógica que permite la interacción entre el front-end y el back-end (servidor), así como el formateo de los datos recibidos y a enviar. A grandes rasgos, actualmente consta de los siguientes archivos con una breve explicación para cada uno:

- **Index.html**: apartado en el que define la interfaz gráfica, que viene a ser únicamente la conversación y un aspecto simplista. Se incluyen todos los archivos javascript del lado del cliente.
- **Aspecto.css**: archivo que incluye todas y cada una de las configuraciones en relación al aspecto gráfico de la aplicación.
- **Common.js**: incluye funciones de uso común en el resto de archivos. Entre ellas está la que permite convertir un JSON en un elemento DOM para poder incluirlo en la conversación con las contestaciones. O un disparador de eventos a partir de lo que se le indique.
- **Api.js**: es el puente entre la parte de cliente y la parte servidor. Da formato a la información que se le va a enviar, vía "HTTP Request", al servidor. También está a la espera de la contestación del mismo. Permite gestionar, también, el uso de las Request y los response para así poder retroceder en la conversación en caso de necesidad o acceder a los datos de contexto de una contestación concreta.
- **Conversation.js**: contiene funciones como la inicialización de la conversación, la gestión de eventos (darle a ENTER, reiniciar la conversación...), así como la construcción de los JSON que serán convertidos a elementos DOM o el desplazamiento automático del chat al último elemento añadido.
- **Payload.js**: tiene como función manejar todo aquello que se muestra y el comportamiento de la columna de la conversación.

3.3.2. Servidor

Es la parte a la que el cliente no tiene acceso directamente, si no que requiere de API RESTful para poder mandar y recibir datos. Dado que hace uso de NodeJS, en este apartado también se hablará de módulos, que vienen a ser como una especie de librerías al estilo C++. Como en el caso del cliente, también consta de una serie de archivos con funciones diferentes:

- **App.js**: punto de entrada de las peticiones del cliente. Es donde se configuran los "endpoints" según la petición recibida. Es desde este archivo desde donde se realiza tanto la conexión a la BBDD como la conexión a la WATSON Conversation. Es básicamente el núcleo del orquestador.
- **Server.js**: simplemente se encarga de poner el servidor a la escucha en el puerto indicado, siendo en este caso el 3000. Cuando recibe alguna petición, app.js lo gestiona.
- **.env**: archivo que simplemente guarda variables de entorno con credenciales como el usuario o el password tanto de la conversation como de la base de datos.
- **Package.json**: archivo en el que se incluyen todos aquellos módulos utilizados en la aplicación, así como el nombre de la aplicación, descripción, etcétera.

3.3.3. Módulos

Los módulos se incluyen en la aplicación como objetos de los cuales se pueden usar los métodos que ofrecen. Actualmente se están utilizando los siguientes:

1. **Módulo EXPRESS**: es un módulo que permite, entre otras cosas, configurar direccionamientos para poder responder frente a diversas peticiones de clientes. Además, los métodos de solicitud pueden ser GET, POST, DELETE, PUT... Para su uso, hay que instanciar express, que en este caso simplemente se lleva a cabo app = express(). Tras ello, se establecen los diferentes "endpoints", así como los parámetros a usar.
2. **Módulo BODY-PARSER**: es usado por express y permite parsear los cuerpos de los request que llegan al servidor.

3. **Módulo CONVERSATION**: módulo que se utiliza para autenticarse e indicar los parámetros de la conversación a la cual deseamos acceder. También se usa para transmitirle los mensajes a Watson y obtener las respuestas. Básicamente hace de nexo entre Watson conversation y el servidor.
4. **Módulo DOTENV**: únicamente carga las variables de entorno presentes en el archivo ".env" en "process.env".
5. **Módulo NANO**: módulo que se utiliza para conectarse con CouchDB y realizar acciones como la creación de bases de datos, su uso, la eliminación, inserción de documentos, etcétera.
6. **Módulo SOAP**: es el módulo que permite la comunicación con el web Service generado mediante Blue Prism. Se le pasan los valores de entrada que dicho WS espera, además de las credenciales de Blue Prism.
7. **Módulo FS**: módulo que permite gestionar y acceder a los archivos del sistema. Es esencial para la interactuación entre el servidor y los archivos locales.
8. **Módulo CLOUDANT**: módulo que permite la conexión con la base de datos que se encuentra en IBM Cloud. Dicha BBDD es no relacional y las funciones a utilizar, obviamente, usan el mismo principio que cualquier BBDD no relacional. Entre otras cosas, son necesarias las credenciales para poder conectarse a la BBDD.

3.4. BLUE PRISM

Actualmente es el encargado de gestionar el envío de correos con el PDF solicitado a Watson. Se ha construido un Web Service que recibe como entrada el correo de destino, el usuario y el contenido. El proceso desarrollado hasta ahora simplemente incluye el envío de correos. Para ello se configura el remitente y el destinatario. En la figura 12 se apreciar el establecimiento de los datos.

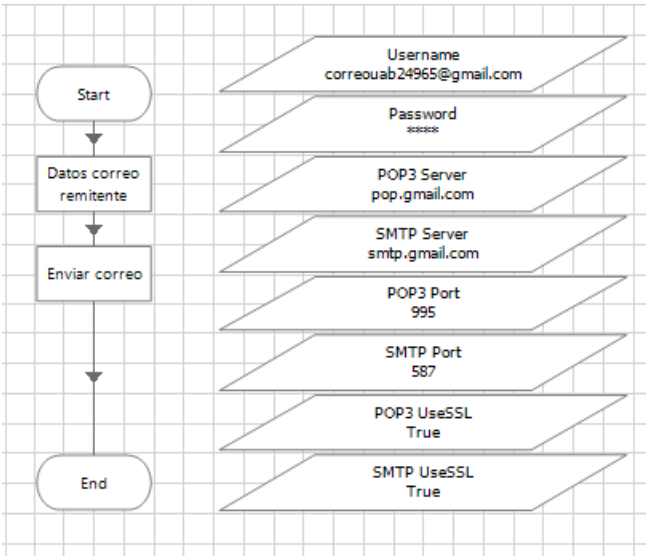

Fig. 12: Configuración del correo

En la figura 13 se ve el flujo del programa desde que entran los datos enviados desde NodeJS hasta que se devuelve el resultado.

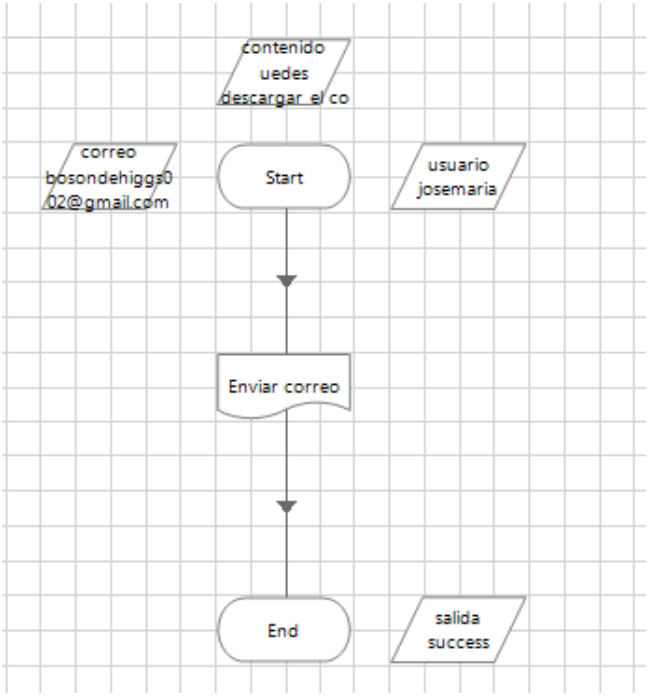

Fig. 13: Muestra la construcción del WS mediante módulos programados en Java

En la figura 14 se aprecia la programación realizada para el módulo de envío de mensajes que se utiliza.

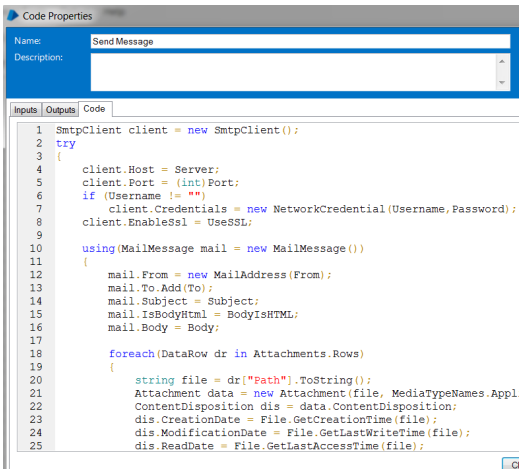

Fig. 14: Muestra el código usado para enviar un mensaje

Todo este proceso, además de unos inputs tiene también un output, que sirve para indicar como ha ido el proceso. Los objetos usados para llevar a cabo las acciones son programados en Java:

Hay que destacar que el software está instalado en local, lo que implica que el ordenador tenga que estar encendido y el software ejecutándose para que se lleve a cabo, en este caso, el envío de correo cuando se le haga la petición mediante *web service*.

4 PROCEDIMIENTO DEL ENTRENAMIENTO

En IBM Cloud, los servicios ofrecidos se clasifican de 2 maneras: pre-entrenados y sin entrenar. Por un lado, los servicios pre-entrenados han sido entrenados por un equipo

de IBM Watson, cuya responsabilidad ha sido obtener los datos correctos para entrenar los modelos de ML y darles a los desarrolladores una funcionalidad "fuera de la caja"para que puedan centrarse en la lógica del software que están desarrollando. Los servicios que vienen pre-entrenados son: Natural Language Understanding, Personality Insights, Tone Analyzer, Speech-to-text, Language Translator y Visual Recognition. Aún así, se puede llevar a cabo un mayor entrenamiento, focalizado en aquello en lo que el desarrollador tenga como objetivo. Por otro lado, están los servicios que requieren entrenamiento de modelos personalizados de ML. Estos servicios son: Natural Language Classifier, Watson Conversation y Visual Recognition. En este caso estamos usando el servicio de Watson Conversation, por lo tanto, va a ser necesario crear un modelo de entrenamiento personalizado. Los siguientes subapartados definen lo que he llevado a cabo.

4.1. ¿Cuánto entrenamiento necesita el Conversation?

Lo primero que hay que saber es hasta dónde hay que entrenar a Watson y en qué estado se encuentra tras cada entrenamiento. Para ello existe un software cuyo nombre es *Watson Developer Cloud Performance Evaluation (DCPE)* [19] que nos va a permitir conocer diversas métricas relacionadas con el entrenamiento. Algunas de estas métricas son:

- **Accuracy (Exactitud)**: de todas las predicciones, cuántas son correctas. Consiste en la suma de las todos los enunciados positivos que Watson ha dado como positivos más todos los enunciados negativos que Watson ha dado negativos. Esto se divide entre el número total de enunciados.
- **Precision (Precisión)**: de todos los enunciados de los cuales se ha predicho que pertenecen a una clase concreta, cuántos pertenecen realmente a esa clase. En este caso, sería el número total de enunciados verdaderamente positivos dividido entre el número total de enunciados predichos como positivos.
- **Recall (Exhaustividad)**: de todos los enunciados que están etiquetados con una clase concreta, cuántos ha predicho correctamente Watson que realmente pertenecen a dicha clase. Es decir, el número total de verdaderos positivos dividido entre el número total de enunciados que están etiquetados como positivos.
- **F1-score**: define la precisión de un test. Hace uso de la precisión (P) y la exhaustividad (E) para poder obtener el resultado: $(2 * P * E)/(P + E)$
- **Matriz de confusión**: relaciona los valores reales y predichos mediante una matriz.

Con todos estos valores se puede ir comprobando cuán bien entrenado está el Conversation y cuándo realmente uno considera que ya ha alcanzado un punto que supera las expectativas establecidas.

4.2. ¿Cómo es el proceso de entrenamiento y evaluación?

En el caso del Conversation, lo primero que va a hacer falta es crear un *groundtruth*, que consiste en un conjunto de datos de entrada (dataset) de ejemplo y su etiqueta (clase) correcta correspondiente. Dicho *groundtruth* se divide de forma aleatoria en 2 conjuntos, uno de entrenamiento y otro de test. Se puede ver un ejemplo de entrada en la figura 15.

```
calendario academico ciencias,calendario
```

Fig. 15: Ejemplo de entrada de groundtruth

Dicha división suele consistir en un 70 % para entrenamiento y un 30 % para test. El conjunto de entrenamiento se usa para construir los modelos personalizados, para ello se sube al servicio de Watson Conversation ya sea vía REST API o el uso de la herramienta interactiva que te ofrecen. Lo que se sube es un archivo en formato *.csv* que en la primera columna contiene los ejemplos y en la segunda columna la clase/intent a la que pertenece. Cuando el modelo ha sido entrenado, se procede a hacer uso de la aplicación referida anteriormente para evaluar el funcionamiento del modelo. En el caso del Conversation, dicha aplicación requiere una serie de datos que se le pasan mediante un archivo JSON:

- La **URL** de la API del Conversation
- El nombre de **usuario** de la Conversation
- La **contraseña** del usuario de la Conversation
- La **ID** del workspace del Conversation
- La **ruta** al archivo de test en formato csv. Este archivo debe tener exactamente el mismo formato que el de entrenamiento.
- El archivo en formato csv donde se guardarán los resultados
- El archivo en formato csv donde se guardará la matriz de confusión

Una vez pasados dichos valores, la aplicación ejecutará un script que se menciona en el siguiente apartado.

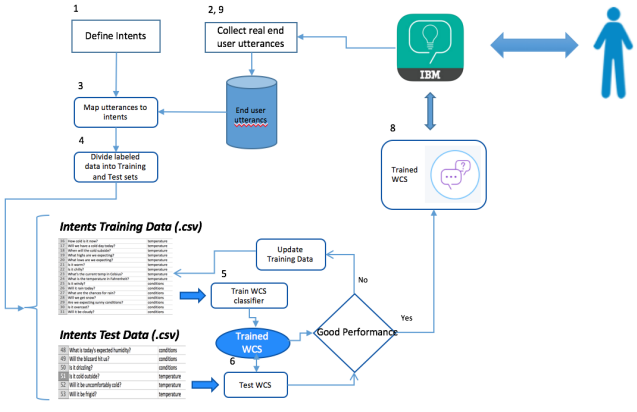

Fig. 16: Diagrama del entrenamiento de *Watson Conversation*

4.3. Pasos seguidos en el entrenamiento y testeo del presente Conversation

El conjunto de entrenamiento consta de un conjunto de frases (o ejemplos) y un conjunto de clases, cada frase se

asocia a una única clase. Se ha aplicado cross-validation (k=4) para comprobar y asegurar la calidad del modelo. A continuación se indican los pasos seguidos para poder llevar a cabo un correcto entrenamiento con el Conversation del Asistente propuesto para la UAB.

Primero se ha definido un *groundtruth* de 500 ejemplos divididos en un total de 7 clases (Intents o Intenciones): **calendario**, **disponibilidad**, **horarios**, **informacion**, **lugar**, **saludos** y **web**. El que sean 7 clases exactamente se debe a una decisión propia y no al hecho de que no puedan haber más, ya que en realidad, a la larga y por la experiencia de trabajar con otros conversations, harán falta decenas de clases. En este caso, debido a la amplitud del proyecto se ha decidido coger únicamente 7. Por otro lado, los ejemplos se han generado teniendo en cuenta diferentes perfiles, intentando emplear diferentes estilos comunicativos: adolescente, estudiante universitario, adultos con diversos estatus sociales, ancianos, etc. Esto ha sido así para poder abarcar la mayor cantidad de *targets* posible. De todos modos, la elección de la cantidad de ejemplos ha sido arbitraria. Por el tamaño que puede llegar a tener el asistente, se requiere de una mayor cantidad de ejemplos, que también, por experiencia, podrían rondar los 250 por clase. El *groundtruth* se ha separado en dos partes: entrenamiento y test.

La parte de **entrenamiento**, que consta de 373 ejemplos con sus respectivas clases, es la que se ha subido al apartado de 'Intents' del Conversation en IBM Cloud, donde automáticamente Watson ejecuta el entrenamiento. Dicho entrenamiento dura apenas 2 minutos. Tras el entrenamiento, Watson ya está listo para recibir peticiones de conversación mediante API Rest. Todo el entrenamiento queda alojado en la nube de IBM. Toda la información que le enviemos a Watson mediante las peticiones se guarda por defecto, tanto la *request* como la *response*. Aún así, indican que la información ni se hace pública ni se comparte, únicamente la utilizan para mejorar el servicio ofrecido. Si no se desea que así se lleve a cabo, hay un parámetro en el header al enviar la petición que es *X-Watson-Learning-Opt-Out*, el cual se ha de poner a **true**.

Para poder ejecutar la parte de **test**, que consta de 127 ejemplos con sus respectivas clases, hace falta el software DCPE mencionado en el apartado 4.1. Dicho software requiere: Git, Python 2 o 3, Anaconda, una cuenta en IBM Cloud y una instancia, en este caso, de *Watson Conversation*.
El proceso a seguir para su correcto funcionamiento se indica en la página de *GitHub* del autor del script [19].

4.3.1. Script de testeo

Para que el script funciones, se ha de clonar el repositorio en el que se encuentra, modificar el JSON correspondiente con los datos descritos en el apartado 4.2, ejecutar el comando *jupyter notebook* en la consola desde el directorio notebooks del repositorio recién descargado y seleccionar el archivo de cálculo de *performance* que coincida con el servicio, 'ConversationPerformanceEval.ipynb' en este caso. Tras ello simplemente se abrirá una página con el código a ejecutar y que consiste en un script escrito en python que se tiene que ejecutar por partes. Toma como referencia los valores indicados en el archivo JSON mencionado en el apartado 4.2. En el script deberán modificarse los siguientes datos para que no de errores ni de versión ni de región, ya que por defecto el script está pensado para Conversations que se han creado en la región de Estados Unidos del Sur y el presente proyecto se ha creado en la región de Alemania:

- el nombre del archivo con los parámetros, indicando aquel que haga referencia a nuestro servicio, siendo en este caso 'example_conversation_parms.json'. En este archivo se encuentran los nombres del csv de test y los csv de salida: matriz de confusión y resultados para cada clase. También contiene los datos de la Conversation.
- la fecha de versión del conversation, en este caso '2016-10-21'.
- la variable 'url' del objeto *conversation*. Se modifica por la URL de la API de la región en la que nos encontremos, en este caso *'https://gateway-fra.watsonplatform.net/conversation/api'* (Alemania)

Algo importante ha tener en cuenta es que los acentos dan problemas con el script. Es por ello que en cualquiera de los casos hago uso de una función que los elimina. Una vez establecidos los parámetros correspondientes, se puede ejecutar el script paso por paso, viendo todas las salidas que nos proporciona cada una de las partes. El script usará el archivo de test pasado con el formato indicado anteriormente y los conjuntos 'ejemplo' y 'clase/intent' separados por ',' y no por ';'. Con él genera el *output* para los archivos de que contendrán la información de la matriz de confusión y los valores de 'precision' y 'recall' entre otros. Con la información devuelta por el script ya se puede estudiar si será necesario o no llevar más entrenamiento sobre Watson.

A grandes rasgos, lo que el script lleva a cabo es:

1. Carga la información del archivo JSON en diferentes variables.
2. Obtiene cada una de las filas del archivo de test, separando el ejemplo de la clase para poder mandárselo a la API de Watson.
3. Realiza la petición para cada una de las frases y guarda los diversos resultados (precisión, exhaustividad...) para poder generar las tablas y matrices. Los resultados también se guardan en los archivos indicados en el JSON.
4. Se imprimen la matriz de confusión, la tabla con las diferentes clases y sus resultados, la precisión...

5 RESULTADOS EXPERIMENTALES

5.1. Etapa 0: Sin INTENTS

Como ya se ha ido indicando a lo largo del dossier, los *intents* básicamente son las clases. Simplemente, IBM ha decidido darles un nombre más descriptivo, que permite hacer entender a los desarrolladores que un *intent* define la intención de un usuario al escribir sobre algo en el chat. Es por ello que sin *intents* Watson no recibe ningún tipo de entrenamiento (ya que como se ha indicado anteriormente, en este caso somos nosotros los que tenemos que entrenarlo, no viene pre-entrenado). Por lo tanto, se escriba lo que se escriba en la conversación, Watson nunca será capaz de conocer las intenciones, ya que la confianza para cada respuesta es de 0, lo que supone dar como respuesta la creada

por defecto. Dicha respuesta consiste en un mensaje indicando que no ha entendido aquello que a lo que hace referencia el usuario. Es por ello que se ha de llevar a cabo un mínimo entrenamiento.

5.2. Etapa 1: Primer entrenamiento

Como se ha indicado anteriormente en el apartado 4.3, para el entrenamiento básico se han usado un total de 7 clases con 500 ejemplos repartidos entre ellas, como se puede ver en la figura 17.

Clase	Nº Training (373)	Nº Test (127)	Total
calendario	50	17	67
disponibilidad	18	7	25
horarios	40	13	53
informacion	62	21	83
lugar	57	19	76
saludos	34	12	46
web	112	38	150
Total	**373**	**127**	**500**

Fig. 17: Cantidades por clase y train/test del primer entrenamiento

La cantidad de ejemplos no es equitativa para cada clase, depende de la cantidad de conceptos que se abarquen. En este caso, como se puede ver gracias a la ejecuciones del script, las clases que más ejemplos tienen son **informacion**, **lugar** y **web**. Las estadísticas mostradas en la figura 18 han sido recopiladas por el propio servicio de Watson Conversation y muestran la cantidad de veces que se ha identificado cada *intent*. Es mayor la cantidad de unas clases que de otras debido a que los ejemplos han sido añadidos por mi y por lo tanto he valorado a criterio personal que tenía que haber más de una cantidad que de otra, lo que produce un sesgo en los datos. Por otro lado, el número total de registros de intents es mayor que el de frases de testeo debido a que el script de Python se ha ejecutado más de una vez, lo que ha supuesto que Watson vaya registrando los intents de varias ejecuciones.

Se han separado en dos conjuntos de 373 y 127, entrenamiento y test respectivamente. Tras llevar a cabo los pasos anteriores y habiendo ejecutado el script, se han obtenido datos bastante buenos para ser un primer entrenamiento y sin separación proporcional. Según la matriz de confusión de la figura 19 obtenida a partir de los datos generados y guardados en el ".csv"de salida a partir del cross-validation, muestra que la mayor cantidad de coincidencias se han dado con **web**, **informacion** y **lugar**. Esto también se debe a lo mencionado, y es que hay más ejemplos para estas clases que para el resto. Se puede apreciar que por lo general las predicciones son bastante buenas. Sobretodo ha habido etiquetados incorrectos en el caso de la clase **informacion**, ya que algunos de los ejemplos comparten una misma estructura inicial pero con diferente desenlace.

En la figura 20 se pueden comprobar varias cosas:

- Tanto con **web** como con **calendario** no ha habido casi ningún conflicto debido a que todos los ejemplos usados establecen una diferenciación clara de la intención del usuario, es decir, es muy difícil considerar una intención diferente a la real.
- En el caso de **saludos** se puede ver que la precisión ha obtenido el máximo valor pero en el caso de la exhaustividad un 0.67. De ahí que salga un *f1-score* de 0.8.
- En lo casos que no hay ido tan bien, como **saludos** o **lugar** (0.83 de *f1-score*) es debido a los pocos ejemplos presentes para llevar a cabo el entrenamiento y el test o a la semejanza entre las preguntas que se realizan. Es por ello que en el segundo entrenamiento se van a añadir más ejemplos para todos los casos en los que escaseen y además los resultados no sean buenos.
- De media se puede comprobar que los resultados son muy buenos, ya que se encuentran entre el 0.8 y el 0.97, lo que indica un alto grado de aciertos por parte de Watson.

#web	226
#informacion	139
#lugar	125
#calendario	118
#horarios	96
#saludos	71
#disponibilidad	48

Fig. 18: Cantidades por cada *Intent*

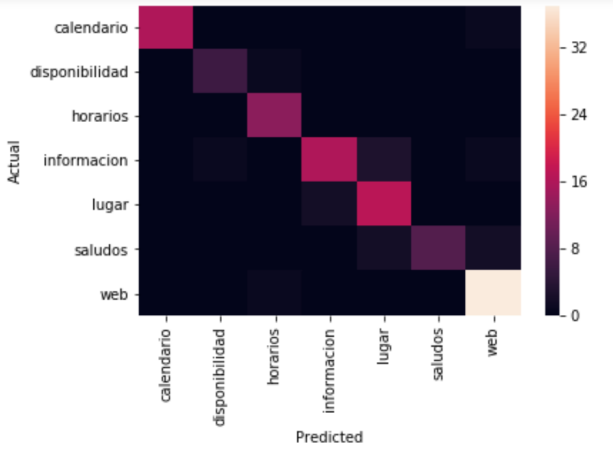

Fig. 19: Matriz de confusión del primer entrenamiento

	precision	recall	f1-score	support
horarios	0,87	1	0,93	13
web	0,9	0,97	0,94	38
calendario	1	0,94	0,97	17
saludos	1	0,67	0,8	12
informacion	0,89	0,76	0,82	21
lugar	0,77	0,89	0,83	19
disponibilidad	0,86	0,86	0,86	7
avg / total	0,9	0,89	0,89	127

Fig. 20: Precisión, exhaustividad (recall), f1-score primer entrenamiento

En este primer entrenamiento se han obtenido una media de $0,92287$ y una desviación estándar de $0,14492$.

5.3. Segundo entrenamiento

En este segundo entrenamiento se ha hecho uso de un total de 600 ejemplos. En la figura 21 se puede ver la distribución.

Clase	Nº Training (450)	Nº Test (150)	Total
calendario	65	21	86
disponibilidad	24	8	32
horarios	47	16	63
informacion	76	25	101
lugar	68	23	91
saludos	43	15	58
web	127	42	169
Total	**450**	**150**	**600**

Fig. 21: Cantidades por clase y train/test del segundo entrenamiento

Se puede apreciar el sesgo en los datos, ya que soy yo quien ciertamente, tras ver qué clases funcionan peor, he decidido cuales requieren más ejemplos. También en este caso se ha usado cross-validation para evaluar los resultados. En este segundo caso podemos comprobar, viendo la figura 22, que los *intents* más usados siguen siendo **informacion**, **lugar** y **web**, ya que siguen siendo los que más ejemplos tienen.

En la matriz de confusión de la figura 23 podremos comprobar que ha mejorado mucho y para todas las clases. Todavía sigue habiendo falsos positivos con la clase **web**, pero es debido básicamente a la multitud de ejemplos que tiene y la terminología empleada, lo que lleva a Watson a dar un confianza similar tanto a la clase **web** como con las que tiene conflicto. De todos modos, la capacidad de Watson para detectar la intención del usuario ha mejorado considerablemente, como también podemos ver en la figura 24, donde podemos ver que para todas las clases la precisión gira alrededor del 0.95, así como la exhaustividad (y consecuentemente el f1-score). Tiene un valor un poco más bajo la clase **lugar**, que es la que lleva a mayor confusión entre ella y **web**.

En este caso se ha calculado la media y la desviación típica de los ejemplos de test, obteniendo unos resultados de $0,94434$ y de $0,13954$ respectivamente.

#web	269
#informacion	173
#lugar	151
#calendario	150
#horarios	125
#saludos	79
#disponibilidad	62

Fig. 22: Cantidades por cada *Intent* usados en el segundo entrenamiento

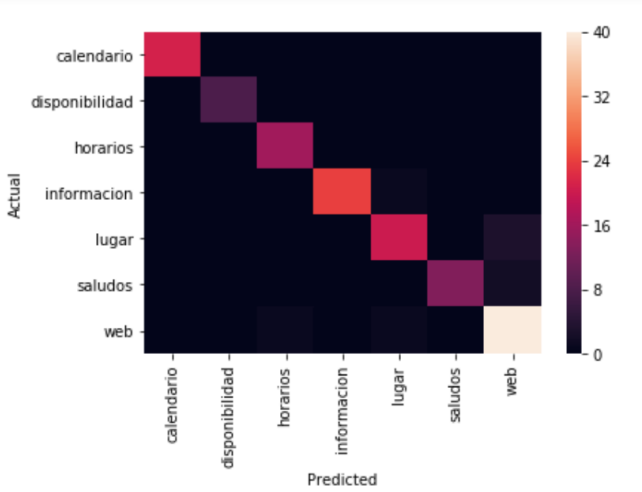

Fig. 23: Matriz de confusión del segundo entrenamiento

6 Conclusiones

Este trabajo fue iniciado con la intención de que tuviese una base de software sólida (nodejs, blueprism), así como una base de computación sólida. A lo largo de la investigación llevada a cabo, se ha visto que, por la parte de computación, el acceso al funcionamiento interno de Watson está prácticamente blindado. Es por ello que se ha tomado otro enfoque: se ha investigado cómo funcionan las herramientas que ofrece este sistema (reconocimiento visual, análisis del habla...), y más concretamente la empleada para este trabajo (Watson Conversation) y qué sistemas utilizan para conseguir sus objetivos; también se ha centrado en la investigación para la obtención de un mejor entrenamiento de Watson, lo que ha permitido llevar a cabo un entrenamiento de un modelo personalizado de ML que funciona muy bien. Esto se debe a la multitud de ejemplos y variantes de una mismo ejemplo que se han empleado para poder obtener un buen *groundtruth*. Así la conversación con Watson es más

	precision	recall	f1-score	support
horarios	0,95	0,95	0,95	16
web	0,92	0,98	0,95	42
calendario	1	1	1	21
saludos	0,97	0,93	0,95	15
informacion	0,96	0,9	0,93	25
lugar	0,94	0,92	0,93	23
disponibilidad	0,94	0,9	0,92	8
avg / total	0,95	0,94	0,94	150

Fig. 24: Precisión, exhaustividad (recall), f1-score del segundo entrenamiento

inteligente y más dinámica, ya que acepta un amplio abanico de expresiones. Por otro lado, la parte de software ha permitido aprovechar la amplia oferta de servicios y la facilidad de interconexión de Watson, lo que conlleva que el usuario pueda hacer peticiones y Watson le entregue aquello que pida: PDF, páginas web, envíos de correo con la información solicitada, etcétera. Todo ello mediante el uso, como ya se ha mencionado a lo largo del dossier, de BluePrism, Nodejs y los servicios de Watson como Cloudant, publicación de aplicaciones y Conversation. Todo esto ha conllevado un entendimiento y comprensión más profundo tanto del funcionamiento del software de computación que ofrece Watson, como así de las herramientas de personalización.

AGRADECIMIENTOS

Quiero expresar mi más sincero agradecimiento a *Fernando Luis Vilariño Freire* por todo el apoyo y la ayuda brindada para el desarrollo de este trabajo, especialmente de la parte computacional. Agradezco de verdad el tiempo que me ha dedicado.

REFERENCIAS

[1] Stanfordnlp.github.io. (2017). Stanford CoreNLP – Natural language software — Stanford CoreNLP. [online] Available at: https://stanfordnlp.github.io/CoreNLP/ [Accessed 19 Sep. 2017].

[2] Google Cloud Platform. (2017). Vision API - Image Content Analysis — Google Cloud Platform. [online] Available at: https://cloud.google.com/vision/ [Accessed 2 Sep. 2017].

[3] media.gm.com. (2017). Hello, OnStar – Meet Watson. [online] Available at: http://media.gm.com/media/us/en/gm/news.detail.html/content/Pages/news/us/en/2016/oct/1025-watson.html [Accessed 27 Sep. 2017].

[4] American Cancer Society MediaRoom. (2017). American Cancer Society and IBM Collaborate to Create Virtual Cancer Health Advisor. [online] Available at: http://pressroom.cancer.org/WatsonACSLaunch [Accessed 18 Sep. 2017].

[5] Shaw, D. (2017). How Wimbledon is using IBM Watson AI to power highlights, analytics and enriched fan experiences - Watson. [online] Watson. Available at: https://www.ibm.com/blogs/watson/2017/07/ibm-watsons-ai-is-powering-wimbledon-highlights-analytics-and-a-fan-experiences/ [Accessed 27 Sep. 2017].

[6] Mullinix, B. (2017). Rocket Fuel Expands Partnership with IBM by Embedding Watson Cognitive Capabilities into Its Predictive Marketing Platform - Rocket Fuel. [online] Rocket Fuel. Available at: https://rocketfuel.com/rocket-fuel-expands-partnership-with-ibm/ [Accessed 2 Oct. 2017].

[7] Yao, M. (2017). 10 Major Brands Using IBM Watson - TOPBOTS. [online] TOPBOTS. Available at: https://www.topbots.com/10-major-fortune-500-brands-using-ibm-watson/ [Accessed 2 Oct. 2017].

[8] KIM, S. (2017). 9 Cool Ways IBM Plans to Make Billions From Watson. [online] ABC News. Available at: http://abcnews.go.com/Business/top-commercial-ibm-watson-jeopardy/story?id=21477280 [Accessed 1 Oct. 2017].

[9] Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A., Lally, A., Murdock, J., Nyberg, E., Prager, J., Schlaefer, N. and Welty, C. (2010). Building Watson: An Overview of the DeepQA Project. AI Magazine, 31(3), p.59.

[10] Burns, S. (2017). Conversation Patterns with IBM Watson – Simon Burns – Medium. [online] Medium. Available at: https://medium.com/@snrubnomis/conversation-patterns-with-ibm-watson-6c4be05e2fe5 [Accessed 14 Sep. 2017].

[11] Burns, S. (2017). Getting Chatty with IBM Watson – Simon Burns – Medium. [online] Medium. Available at: https://medium.com/@snrubnomis/getting-chatty-with-ibm-watson-1075c549ee9e [Accessed 14 Sep. 2017].

[12] Ferrucci, D., Levas, A., Bagchi, S., Gondek, D. and Mueller, E. (2013). Watson: Beyond Jeopardy!. Artificial Intelligence, 199-200, pp.93-105.

[13] Medicina basada en hechos, https://es.wikipedia.org/wiki/Medicina_basada_en_hechos

[14] Bakkar, N., Kovalik, T., Lorenzini, B., Spangler, S., Lacoste, A., Sponaugle, K., Ferrante, P., Argentinis, E., Sattler, R. and Bowser, R. (2017). Artificial intelligence in neurodegenerative disease research: use of IBM Watson to identify additional RNA-binding proteins altered in amyotrophic lateral sclerosis. Barrow Neurological Institute.

[15] Tsoi, K., Chan, F., Hirai, H., Keung, G., Kuo, Y., Tai, S. and Meng, H. (2017). Data Visualization with IBM Watson Analytics for Global Cancer Trends Comparison from World Health Organization. International Journal of Healthcare Information Systems and Informatics, 13(1), pp.45-54.

[16] Chozas, A., Memeti, S. and Pllana, S. (2017). Using Cognitive Computing for Learning Parallel Programming: An IBM Watson Solution. Procedia Computer Science, 108, pp.2121-2130.

[17] Karpov, A. (2017). 32 OpenMP traps for C++ developers — Intel® Software. [online] Software.intel.com. Available at: https://software.intel.com/en-us/articles/32-openmp-traps-for-c-developers [Accessed 22 Dec. 2017].

[18] Singh, S. and Rawat, M. (2017). Design and Analysis of Visual Insight of Child using IBM Multimedia Analysis and Retrieval System. International Journal of Computer Applications, 161(6), pp.19-21.

[19] Kozhaya, J. (2018). joe4k/wdcutils. [online] GitHub. Available at: https://github.com/joe4k/wdcutils/ [Accessed 5 Jan. 2018].

APÉNDICE

A.1. Discovery

Fig. 25: Arquitectura de Discovery

Como se puede apreciar en la imagen, Discovery "crawlea", convierte y enriquece los documentos pasados, ya sean JSON, HTML, etc. Les añade metadatos NLP (Natural Language Understanding) que facilitan la exploración y obtención de ideas. Posteriormente son limpiados y normalizados para mejorar la calidad de los datos. Una vez llevado a cabo, se indexa en una colección que pasa a formar parte del entorno, lo que facilita la obtención de hipótesis, la interpretación de los datos... Esto lo aprovecha la aplicación para poder obtener de forma sencilla ideas o conceptos sobre un determinado conjunto de datos sin estructura.

A.2. Knowledge Studio

Para llevar a cabo lo mencionado en Knowledge Studio del apartado 2.2.2, sigue una serie de pasos:

1. Importar los documentos de origen, a partir de los cuales se creará un sistema de tipos, que consiste en la definición y la relación de estos tipos para la información de interés de la aplicación que vaya a usar este modelo.
2. Una o varias personas clasifican las palabras de uno o varios documentos mediante etiquetas, por ejemplo, un perro es una mascota, una calle es una localización. También han de identificar tipos de relaciones y correferencias.
3. A partir del *Ground Truth*, Watson entrena un modelo.
4. Tras entrenarlo, Watson lo aplica a nuevos documentos para encontrar entidades, relaciones y correferencias.

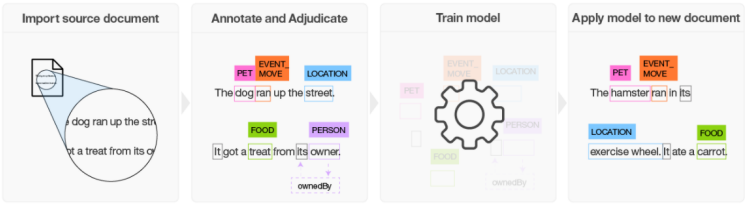

Fig. 26: Diagrama del proceso de WKS

A.3. Language Translator

Para que el Language Translator funcione, deben llevarse a cabo dos pasos:

1. Añadir un glosario propio con palabras o frases en la lengua origen y en la lengua objetivo.
2. Subir una gran cantidad de texto en la lengua objetivo para que sirva como muestra. De esta manera se entrena y a la hora de llevar a cabo la traducción, ésta será mucho más precisa a la par que más elaborada.

A.4. Natural Language Classifier

Para hacer funcionar el NLC primero ha de entrenarse mediante datos de ejemplo, y así poder posteriormente clasificar textos con los cuales no ha sido entrenado. Un ejemplo de uso puede ser en un asistente virtual, ya que puede ser capaz de detectar las intenciones del usuario a medida que va escribiendo y ser capaz de redirigirlo a un departamento específico.

Fig. 27: Diagrama del proceso del NLC

A.5. Natural Language Understanding

Una lista de las posibilidades que ofrece este servicio:

- **Categorías**: categoriza el contenido haciendo uso de una jerarquía de clasificación de 5 niveles basándose en una serie de categorías ya preestablecidas en Watson.
- **Conceptos**: identifica conceptos de "alto nivel" que no están referenciados de forma directa en el texto pasado.
- **Emociones**: identifica las emociones a partir de la terminología utilizada, detectando también el objeto al cual están dirigidas dichas emociones.
- **Entidades**: detecta lugares, personas, eventos... todo basándose en una serie de tipos y subtipos ya preestablecidos en Watson.

- **Palabras clave**: identificación de palabras clave en el texto.
- **Metadatos**: obtención de metadatos de la URL o HTML pasado en el texto.
- **Relaciones**: identifica aquellas entidades que están relacionadas e identifica el tipo de relación.
- **Roles semánticos**: parsea oraciones de tal manera que extrae sujeto, verbo y objeto. Además de identificar tanto entidades como palabras clave que son el sujeto u objeto del verbo (acción).
- **Sentimiento**: Analiza el sentimiento expresado en una frase o incluso en un documento completo.

A.6. Personality Insights

Personality Insights se basa en tres modelos:

1. **Big Five**: afabilidad, conciencia, extraversión, rango emocional y apertura.
2. **Necesidades**: excitación, harmonía, curiosidad, ideal, cercanía, expresión personal, libertad, amor, practicidad, estabilidad, desafío y estructura.
3. **Valores** (que motivan la toma de decisiones): auto trascendencia / ayudar a los demás, conservación / tradición, hedonismo / disfrutar de la vida, mejorarse a uno mismo/ lograr el éxito, abierto al cambio / emocional.

También existe la posibilidad de que detecte preferencias de consumo, lo que estaría más orientado a redes sociales u otras plataformas de ocio. Con todo, PI permite entender la personalidad de las personas, aprender de ellas y adaptarse, por ejemplo, mediante la mejora de los servicios que se estén ofreciendo.

A.7. Speech to Text

Funciona de tal manera que cuanto más audio escucha, mejor es la interpretación de lo que se transcribe. Soporta WebSockets, HTTP REST y HTTP asíncrono. Como entrada puede recibir audios en formato MP3, WAV, FLAC, entre otros. También audio en streaming, hasta 100MB. Como salida puede dar:

- Los diferentes participantes en la conversación
- Palabras clave
- Diferentes transcripciones debidas a diferentes interpretaciones
- Diferentes palabras o confianzas
- Filtrar palabras que no se deseen (insultos...)
- Conversión de formatos: fechas, números de teléfono, números sin más, valores de moneda...

A.8. Text to Speech

En este caso, hace uso de HTTP REST y WebSockets. Las capacidades que ofrece son:

- Diferentes formatos de audio
- Diferentes voces, tonos y lenguas
- Acepta SSML (Speech Synthesis Markup Language)
- Permite añadir un matiz de expresividad al SSML
- Permite añadir transformación de voz al SSML
- Personalización en la pronunciación de palabras

A.9. Tone Analyzer

El proceso del Tone Analyzer consiste en 4 simples pasos:

1. Se sube un documento JSON, HTML o en texto plano (máximo 128KB)
2. Watson lo analiza
3. Devuelve un JSON con todo el análisis llevado a cabo
4. Mediante la información obtenida se reescribe el mensaje para adaptarlo a nuestras intenciones reales

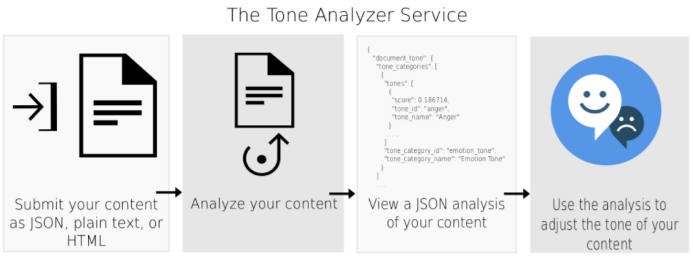

Fig. 28: Diagrama del proceso del TA

A.10. Visual Recognition

Por defecto, Watson ya viene entrenado con multitud de imágenes que los desarrolladores han escogido. Si se quiere conseguir un analizador de imágenes personalizado, con diversas clases personalizadas, se ha de subir un zip con imágenes por cada clase. Es decir, si queremos que detecto un Ford fiesta, se ha de crear un zip con el nombre "Ford fiesta" que incluya imágenes de este vehículo únicamente. Entonces Watson se entrena con estas nuevas clases e imágenes. Hay que tener en cuenta que a la hora de subirlos se tendrá que indicar tanto la clase como si son positivos o negativos. Es decir, si estamos buscando tipos de coche, subiremos como negativos imágenes de motos.

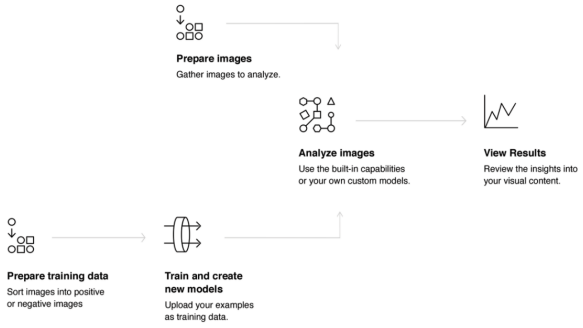

Fig. 29: Diagrama del proceso del VR