UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Bibliotecas de interacción humano chatbot

INFORME DE ACTIVIDADES PROFESIONALES

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Héctor Ricardo Murrieta Bello

ASESOR DE INFORME

Dr. Iván Vladimir Meza Ruiz

Ciudad Universitaria, Cd. Mx., 2018

Agradecimientos

A mi familia, amigos, y maestros
que muchas veces son lo mismo
gracias.

Índice general

Capítulo 1

Introducción

Una de las aplicaciones del procesamiento del lenguaje natural, es el desarrollo de agentes conversacionales, que también reciben el nombre de sistemas de diálogo o chatbots. Durante mi trayectoria como ingeniero en software con especialidad en ingeniería de datos, en la empresa *mariachi.io* desarrollé bibliotecas de interacción humano chatbot. Su utilidad está en implementar metodologías para la creación de sistemas de diálogo.

En el presente informe profesional explico una de las actividades principales que desarrollé en la empresa *mariachi.io*, que fue seleccionar las metodologías correctas, mostrarlas a los otros equipos de desarrollo e implementar herramientas que resuelven los retos en la parte de procesamiento de lenguaje natural dichos objetivos se ven respresentados en bibliotecas para interacción humano chatbot.

A continuación se presentan los objetivos y la estructura del informe profesional.

1.1. Objetivos

Los objetivos de las bibliotecas de interacción humano chatbot giran al rededor de la necesidad que tenía la empresa *mariachi.io* de tener herramientas para desarrollar agentes conversacionales. Dichas herramientas debía contar con las características de ser modificables, escalabes, y especializadas en el español mexicano de la zona del altiplano.

Además desarrollé mecanismos escalables para implementar metodologías de creación de chatbots. Esta ahora usa los equipos de *back-end* pues los agentes que desarrolla *mariachi.io* se han vuelto mas robustos y requieren un mayor número de desarrolladores.

1.2. Estructura del presente informe

En esta sección muestro la forma en la que se organiza el presente reporte. En la sección *2.Mariachi.io* hablo sobre la empresa en la que trabajé que es *mariachi.io* y también describo el puesto en el que me he desarrollado. Éste puesto es el de ingeniero de software con especialidad en ingeniería de datos.

La sección *3.Antecedentes* muestra las ideas que dieron pie al trabajo que desarrollé en la empresa *mariachi.io*. Comienzo dando una breve historia del desarrollo de agentes conversacionales en el mundo. Posteriormente introduzco los conceptos de metodologías de agentes finítos y basados en *frames*.

Posteriormente en la sección *4.Propuestas para interacción humano chatbot* abordo las técnicas que propongo para implementar dos

metodologías. Las basadas en estados finitos y las basadas en *frames*. También muestro cómo las bibliotecas de interacción humano chatbot son una propuesta de valor para implementar esta metodología.

En *5.Metodología* muestro las técnicas y procedimientos que utilicé para desarrollar las bibliotecas de interacción humano chatbot. Éstos procedimientos están basados en técnicas usadas en procesamiento de lenguaje natural y aprendizaje automático.

Después en la sección *6.resultados* hablo sobre algunos ejemplos de proyectos en los que se han utilizado las bibliotecas de interacción humano chatbot. También describo la forma en que los equipos de *mariachi.io* han escalado las propuestas que hice para implementar metodologías de desarrollo de agentes conversacionales.

Por último en la sección *7.Conclusiones* sintetizo las ideas principales de la propuesta y de este informe. Además se exponen problemáticas a resolver y posibles futuras soluciones para las bibliotecas de interacción humano chatbot.

De esta forma describo el trabajo que realicé para la empresa *mariachi.io*. Y la forma en la que se satisfacen los requerimientos de los clientes de la empresa.

Capítulo 2

Mariachi.io

La empresa para la que desarrollé las bibliotecas de interacción humano chatbot es *mariachi.io*. En este capítulo presento la misión y visión de la empresa *mariachi.io*. Además, hago una descripción del puesto que he ocupado desde septiembre de 2016.

Mariachi.io es una empresa joven que lleva más de dos años en el mercado. La especialidd de la empresa es el desarrollo de productos que implementan inteligencia artificial, ya sea en aplicaciones móviles, aplicaciones web, o dispositivos de internet de las cosas.

- **Visión** Mejorar la experiencia de vida de los seres humanos a traves de herramientas que utilicen inteligencia artificial. En el futuro las personas utilizarán éstas herramientas en su vida diraria.

- **Misión** Crear productos inteligentes que faciliten la vida de los seres humanos usando elementos interdisciplinarios. Por un lado se usarán elementos de ingeniería en computación como

aplicaciones móviles, desarrollo web e inteligencia artificial. Por otro se buscará incluir elementos de lingüistica, diseño gráfico, y ciencias cognitivas.

El puesto en el que me he desarrollado es el de ingeniero de software con especialidad en ingeniería de datos. En un periodo de septiembre de 2016 como medio tiempo hasta junio de 2017, y como tiempo completo desde enero de 2018 hasta la fecha.

Las tareas del puesto son: generar soluciones basadas en ingeniería de datos o inteligencia artificial a problemas de visión por computadora o procesamiento de lenguaje natural. Estas soluciones van desde plantear metodologías, programar herramientas, o plantear arquitecturas que mejoren las soluciones.

Una de las soluciones más importantes en las que trabajé, y que es tema central de este informe, son las bibliotecas de interacción humano chatbot. Sin embargo también diseñé soluciones para tareas de visión por computadora.

Al trabajar en una empresa pequeña, parte de las tareas que se esperan del puesto incluyen: levantar requerimientos con los clientes, y ayudar a encontrar clientes. Sin embargo en el presente informe muestro el desarrollo de un producto que se ha vuelto núcleo de la empresa *mariachi.io*.

Duarante mi estancia en la empresa *mariachi.io*, he utilizado los conocimientos de varias materias como: *Ingeniería de software, sistemas operativos, estructuras de datos, estructuras discretas, inteligencia artificial y procesamiento de lenguaje natural*.

.

Capítulo 3

Antecedentes

En el presente capítulo se busca mostrar el contexto en el que se desenvuelve el proyecto en el que trabajé en la empresa *Mariachi.io*, que consistió en el diseño de bibliotecas para interacción humano chatbot. En la sección 3.1 respondo la pregunta ¿Qué son los sistemas de diálogo? y doy un pequeño marco histórico sobre los mismos. Mientras que en la sección 3.2 muestro un panorama sobre las metodologías que funcionaron como base para desarrollar las bibliotecas de interacción humano chatbot.

3.1. Sistemas de Diálogo

Los agentes conversacionales, sistemas de diálogo o chatbots son programas de computadora que busca tomar parte en una conversación humano-computadora. Estos agentes usan el lenguaje natural para formar parte del diálogo. Para esto existen diferentes interfaces: la voz (sistemas de diálogo hablados), el texto y otros hibridos que

conmbinan éstas habilidades, como se ve en los chatbots de Facebook Messenger. Todas estas interfaces usan manejadores de diálogo que buscan tener el control de los diversos elementos en los agentes conversacionales para poder mantener la conversación [2].

En los medios e industria han existido chatbots de muchos tipos y para muchas tareas. El primero del que se tiene registro es ELIZA [13]. Con ELIZA el autor buscaba simular la charla entre un paciente (humano) y un terapeuta de la psicoterapia centrada en el cliente. Este tipo de conversaciones tiene un sentido muy particular pues los terapeutas buscan hacer que los clientes expliquen sus sentimientos, de esta forma el terapeuta no debe mostrar conocimiento de la causa de los problemas o dar soluciones, simplemente hacer las preguntas necesarias para que el cliente pueda explicar a detalle sus problemas y así encontrar las soluciones[11].

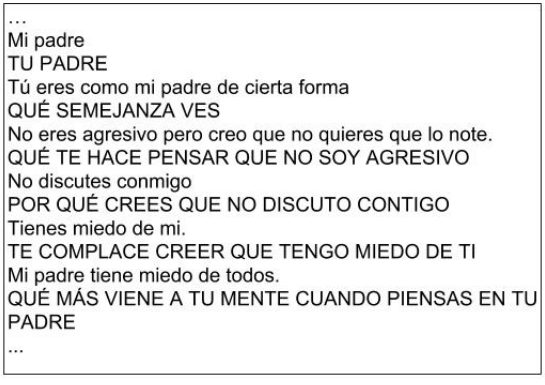

Figura 3.1: Extracto de conversación con ELIZA. Las oraciones en mayúsculas corresponden a respuestas de ELIZA, traducido de [8]

La clave para el desarrollo de ELIZA fue el uso de reglas preestablecidas por Weizenbaum, que modificaban el texto de entrada para

que la conversación tuviera coherencia. De ésta forma a oraciones como *I am sick* ELIZA responde con *Do you believe you are sick*. En la figura3.1 podemos ver un fragmento de una de las charlas más famosas de ELIZA.

Varios años después de ELIZA, surgió A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) un proyecto en Internet que continúa con la idea de seguir usando reglas y patrones para hacer chatbots. Lo interesante en A.L.I.C.E. es que integra AIML(Artificial Intelligence Markup Language), un dialecto basado en XML que fue desarrollado por el Dr.Richard Wallace que incluso hoy en día sigue teniendo mantenimiento por el proyecto Pandorasbot[1].

Actualmente es utilizado pues el proyeto Pandorasbot permite tener portabilidad en multiples pataformas como Facebook Messenger, Telegram, Slack o Twitter. La perspectiva de usar agentes conversacionales para temas en éspecifico ha tenido gran auge en industria, en la sección 3.2 se hablará con más detalle sobre la inclusión de éste paradigma en la industria.

Por otro lado una propuesta distinta es la de estimar la mejor respuesta basada en un corpus de preguntas y respuestas. El objetivo de éstos sistemas es poder establecer conversaciones de temas variados, por lo que no es necesario establecer reglas o patrones a seguir sino que haciendo uso de conversaciones pasadas se logra obtener una respuesta. Algunos autores se refieren a estos sistemas como basados en corpus [7, 8]

Uno de los primeros chatbots en hacer uso de ésta técnica fue MegaHAL[6], que al igual que los agentes modernos puede dividir su

[1]https://home.pandorabots.com/es/ febrero 2018

procesamiento en dos partes. La primera donde se busca procesar la entrada y la segunda que genera una respuesta. Lo que hacía era usar dos modelos ocultos de markov: el primero predice la palabra predecesora de alguna secuencia y el segundo modelo predice la siguiente palabra dadas las palabras anteriores. Con el primer modelo intentaba encontrar las palabras correctas para poder inicializar el segundo algoritmo.

La idea de usar corpus para poder tener agentes que puedan mantener la conversación en un dominio abierto de temas siguió evolucionando. Algunos sistemas como cleverbot[3] o Microsoft's 'XioaIce' buscan aproximar la entrada del usuario a alguna pregunta del corpus y regresa la respuesta de ésta pregunta al usuario. Aunque esta idea ha sido muy popular en investigación por el reto que representa y hemos visto grandes avances haciendo uso de redes neuronales y modelos de secuencia a secuencia, no es sujeto de estudio en éste informe pues no ocupé esa perspectiva en mi estadia en la empresa *mariachi.io*.

3.2. Metodologías modernas en industria

En la industria vemos agentes conversacionales que buscan cumplir objetivos claros a través del diálogo, responder llamadas en un coche, poner música con Amazon Alexa, o agendar un evento con Siri. Esta perspectiva ha sido muy estudiada [8, 2] y es en este tipo de chatbots donde este informe tiene un mayor aporte. Los agentes que buscan resolver tareas en específico han recibido el nombre agentes orientados a tareas.

Cuando se desarrollan estos sistemas, las conversaciones se mo-

delan para poder obtener del usuario la información necesaria para poder ejecutar una acción. Por ejemplo, si lo que se busca es vender un coche, es necesario saber el modelo, el color, y la marca, entre otras cosas. Algunos autores, hablan de dos metodologías clave para poder obtener ésta información, la que está basada en estados finitos y la que usa frames [8, 2].

Dentro del ámbito industrial existen otras propuestas como las basadas en procesos ocultos de markov. También existen iniciativas para generar texto con redes neuronales. Sin embargo esas perspectivas están fuera del alcance de éste informe pues los requerimientos de nuestros clientes no las requirieron.

3.2.1. Agentes basados en estados finítos

Para poder entender la propuesta de los agentes de esta sección tanto de los agentes de la sección 3.2.2 se debe hablar del concepto de turno en una conversación. Un turno en una conversación es una frase que puede ser dicha por el sistema o por el humano [8], en cada estado de la figura 3.3 hay dos turnos. De esta forma, un diálogo está conformado por diferentes turnos. Es común diseñar a los chatbots para que los turnos del agente conversacional se optimicen, de tal forma que se logra llegar al objetivo en la menor cantidad de pasos posibles.

Los agentes basados en estados finitos tienen una estructura fija para obtener la información del usuario, pues los turnos se modelan como estados en un autómata finito. De esta forma en cada estado se sabe qué se va a decir y cuál debe ser el siguiente estado dependiendo de la entrada del usuario. Así el agente tiene un mayor control de la

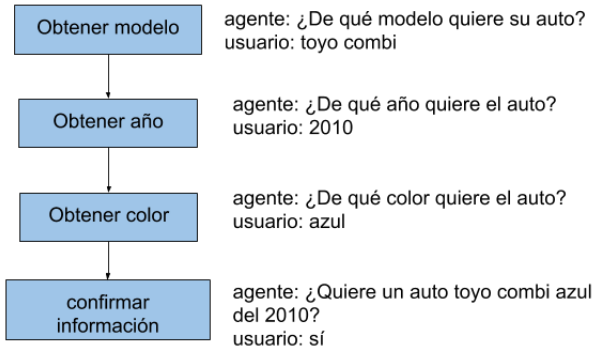

Figura 3.2: Estados muestran el flujo para vender automovil

conversación para poder lograr el objetivo.

En la figura 3.3 se ejemplifican los estados que debe seguir la conversación para poder vender un automovil. Las preguntas se hacen de forma directa y se verifica la información con la que responde el usuario.

Las ventajas que tienen estos sistemas son siguientes:

- Son agentes relativamente mas fáciles de programar
- Como se sabe el tema y se tiene gran noción de la respuesta del usuario es fácil procesar la respuesta
- Bueno en tareas que no son tan complejas, o que deben desarrollarse rápido.

Las desventajas que tienen los agentes basados en estados:

- En el dialogo real, el control no lo tiene solo una de las partes.
- Si el usuario responde con varios elementos del objetivo el sistema solo toma en cuenta el elemento por el que preguntó.

- Las charlas requieren más turnos para poder llegar al objetivo

3.2.2. Agentes basados en *frames*

Como podemos ver la optimización de los turnos y la naturalidad en una conversación son temas de gran relevancia en el proceso de diseño un sistema de diálogo. Una de las respuestas más usadas para poder tener charlas más fluidas es usar *frames*. Los *frames* están formados por bloques o *templates*. Éstos elementos tienen slots definidos por los recursos que se tienen que obtener de la conversación.

En el ejemplo de la sección pasada queríamos vender un coche, cada venta es un *template* o bloque y los *frames* tienen los siguientes *slots* a llenar: el modelo, la marca, el color y el año. Éstos elementos se intentan obtener de las respuestas que da el usuario en cada turno, así cada respuesta puede tener uno, varios o ningun elemento del *frame*.

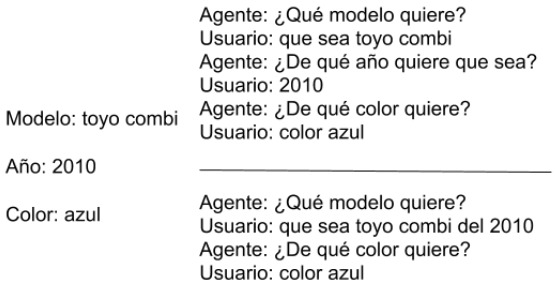

Figura 3.3: Dos conversaciones que usan *frames* para obtener datos

De ésta forma el flujo de la conversación está dado por los elementos que hacen falta llenar para poder completar el frame. En la figura 3.3 podemos ver una charla de un agente que se diseña con la

técnica basada en ontologías. Como podemos ver el numero de turnos se reduce, y los flujos que puede tener la conversación son mucho mas variados.

Las ventajas de usar *frames* son las siguientes:

- El usuario puede responder con varios elementos en un solo turno.
- La charla es más natural.
- El número de turnos necesarios para completar una conversación puede reducirse significativamente.

Las desventajas que tienen éstos sistemas son:

- Puede llegar a ser complicado hacer el sistema para extrar los elementos de los frames.
- Es más dificil de escalar.
- Se deben saber los temas de los que se va a hablar.

El sistema que diseñé para la empresa *Mariachi.io* consiste en bibliotecas que se usan en chatbots. Que pueden utilizarse en chatbots basados tanto en estados finitos como en *frames*.

Capítulo 4

Propuestas para interacción humano chatbot

En el presente capítulo explicaré la importancia de hacer bibliotecas de interacción humano chatbot exclusivas para *mariachi.io* y la problemática que esto implica. Además, mostraré las otras opciones en el mercado que resuelven el problema con metodologías similares.

Las respuestas de otras compañias buscan hacer simple la tarea de diseño de sistemas de diálogo, ya sea en escalabilidad, portabilidad o desarrollo de inteligencia artificial. Dado que son productos, tienen un costo que debe ser cubierto por los desarrolladores o los clientes que pagan por tener los sistemas de diálogo. En la sección 4.1 profundizo en el aporte que tienen éstas propuestas a los desarrolladores y que optan por usar las herramientas que venden.

En el capítulo pasado hablamos sobre la metodología basada en *frames*. Podemos abstraer ésta metodología usando dos conceptos, que han sido usados por muchas compañias en la industria. El primero son

los *intents* que sirven como mecanismo para poder saber de qué frame se está hablando por ejemplo la frase *roomie quiero prender el foco de la sala* se tiene la intención o el *intent encender foco* pues la tarea que debemos cumplir es encender un foco. El segundo son las *entities* o entidades que logran extraer del texto del usuario los elementos que llenan los *slots* de los *frames*, retomando el caso de la oración *roomie quiero prender el foco de la sala* la entidad necesaria es el lugar, en éste ejemplo la entididad de tipo lugar es *sala*.

4.1. Propuestas de otras compañías

Podría hablar de un gran número de empresas que se dedican a crear tecnología para hacer chatbots. En esta sección hablo en específico de tres: *wit.ai*, *chatfuel*, e *IBM Watson*. Las tres empresas tienen la característica de ofrecer sus herramientas como servicios *on demand*, lo que hace que cobren a su cliente por la cantidad de peticiones que le hacen a sus herramientas.

4.1.0.1. *wit.ai*

Wit.ai es una empresa que vende una herramienta para poder hacer agentes conversacionales basados en tareas que intentan modelar su agente basándose en *intents* y *entities*. Funciona de la siguiente manera: los usuarios se registran en su plataforma, pueden crear aplicaciones que básicamente son elementos donde pueden entrenar un algoritmo para poder extraer entidades y clasificar el texto en diferentes

[0]https://wit.ai/ en febrero 2018

entities, con éstos recursos se pueden implementar agentes conversacionales con metodologías tanto basadas en estados finitos como en *frames* .

La forma en la que los usuarios pueden acceder a los recursos de wit.ai, es a través de peticiones a un servidor, estas peticiones llevan la información del texto que quieren procesar y el servidor responde con la intención y las entidades que se obtivueron al procesar el texto enviado por el usuario.

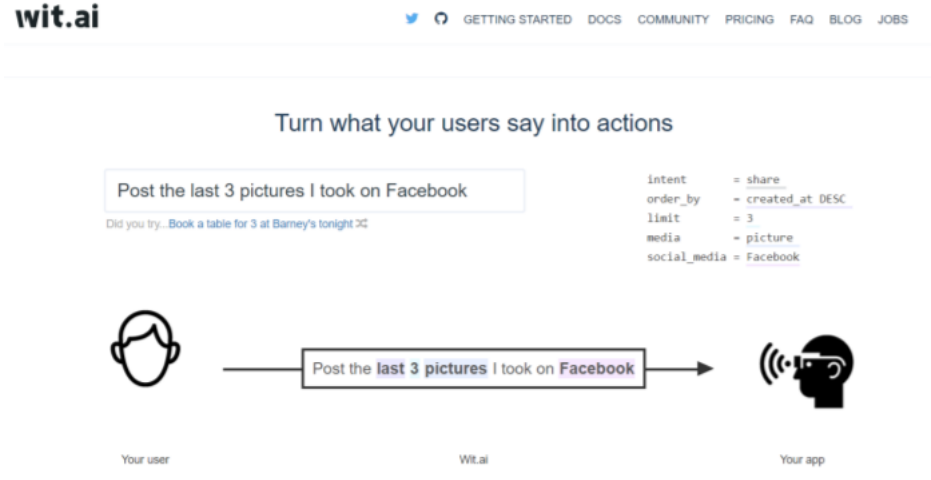

Figura 4.1: Ejemplo de extracción de entidades e intención con *wit.ai* tomado como captura de pantalla de el sitio de *wit.ai*[2]

Entre los aspectos más importantes de wit.ai es que es gratuito, se puede manejar desde una plataforma en línea y funciona también para el español. Las desventajas de este sistema es que el español funciona muy bien para el español de España pero no es tan bueno para el español de México, y al no tener control sobre los algoritmos

que usan, si empiezan a existir errores la única solución es brindar más ejemplos a la plataforma.

4.1.0.2. *Chatfuel*

Chatfuel [3] propone una solución distinta para poder hacer sistemas de diálogo. La propuesta es usar la metodología de agentes basados en sistemas finitos, pues lo que se puede hacer es establecer los flujos que puede seguir la conversación. Puede parecer que es un sistema muy simple, pues la complejidad en la solución que ofrece es mínima, sin embargo en ésta simpleza radica su gran éxito. Para usar la herramiento no es necesario programar, sino que con su interfaz se puede describir todo el flujo de la conversación usando bloques que representan los diferentes elementos que propone Facebook Messenger para hacer chatbots.

Ésta herramienta tiene grandes clientes, como TechCrunch, Adidas, Buzzfed, MTV, y UBER. El gran éxito se debe a que gracias los elementos que se pueden usar en un agente conversacional de Facebook Messenger, diseñar la experiencia de usuario se ha vuelto cada vez más importante. Por otro lado la plataforma también ofrece *analytics* cuando se contratan los servicios mas caros, lo que hace que si el objetivo de desarrollar el agente es pocisionar la marca o tener más visibilidad, chatfuel puede ayudar en gran medida.

[3] https://chatfuel.com/ en febrero 2018

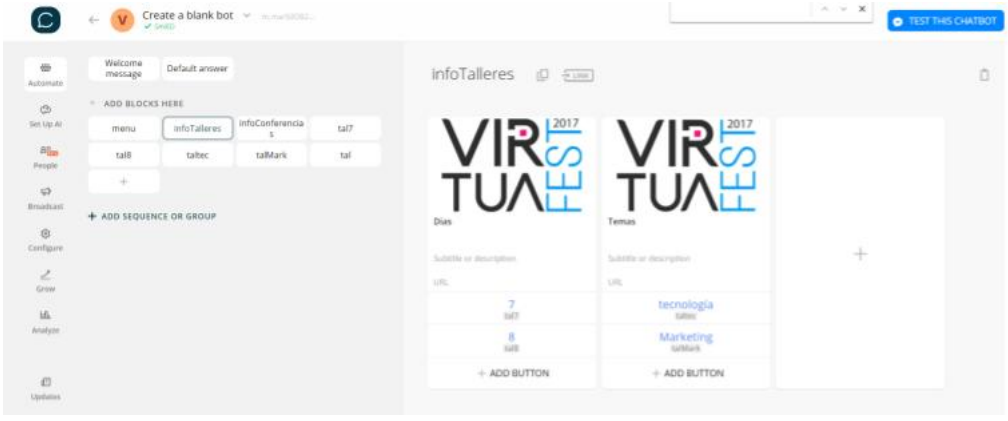

Figura 4.2: *Dashboard* de un chatbot con chatfuel que utiliza elementos de *Facebook Messenger* para tener mejor desempeño tomado de el sitio de chatfuel [5]

4.1.0.3. *IBM Watson*

IBM Watson es una herramienta con diferentres herramientas de usos variados, el sistema que cubre nuestro interés al hablar de sistemas de diálogo es *Watson Conversation*[6]. Ésta suite da una gran propuesta a sus clientes, pues al igual de *wit.ai* también da una solución de *entities* e *intents* para modelar el entendimiento del lenguaje natural, cuanta con elementos para controlar flujos cons sistemas basados en estados finitos y en *frames*, ademas de que puede administrar la portabilidad del sistema en diferentes plataformas como Facebook Messenger, Telegram, y Slack.

Aunque *Watson Conversation* es un sistema muy completo, que ofrece una solución bastante completa en la tarea de hacer sistemas conversacionales basados en objetivos para la industria, mariachi.io

[6]https://www.ibm.com/watson/services/conversation/

decidió no usarlo en sus sistemas por dos razones básicas: la primera la falta de control en los algoritmos (que es un problema en todas las soluciones desarrolladas por teceros) y la segunda es que es una solución con costo mensual, lo que hace que los clientes de *Mariachi.io* deban absorber el pago mensual de la plataforma, y si lo que *mariachi.io* tiene como menta en la consultoría es optimizar los recursos de los clientes, brindar soluciones que no necesiten costos por mantemiento de la inteligencia artificial es una mejor propuesta.

4.2. Propuesta para mariachi.io

En ésta sección describo cuál es el proceso que sugerí para abstraer las metodologías de desarrollo de agentes conversacionales basadas en *frames* o en estados finitos. Es importante resaltar que aunque desarrollar flujos de la conversación y levantar requerimientos de los clientes es parte de mi trabajo, en ésta sección y la siguiente describo el proceso de crear agentes conversacionales después de tener los flujos detallados.

Durante el tiempo en el que he trabajado en la empresa mariachi.io, desarrollé diferentes tareas de consultoría a clientes usando agentes conversacionales. En los proyectos, la propuesta se modificó ligeramente, sin embargo a través de las iteraciones, desarrollé una propuesta para poder crear sistemas de diálogo que es escalable y rápida de generar. De ésta forma *Mariachi.io* puede acelerar los procesos de desarrollo y enfocarse mas en la experiencia de usuario, los flujos de la conversación, desarrollar bases de datos para almacenar información importante para los clientes o incluso enfocarse en atraer

mas clientes a la empresa.

4.2.1. Modelando agentes basados en estados finitos

Cuando se desarrolla un agente conversacional, que va a tener múltiples usuarios es necesario desarrollar un mecanismo que ayude a modelar en qué parte de la conversación de encuentra el usuario, y qué elementos se han obtenido de la conversación para cumplir el objetivo. En la figura4.3 podemos ver un pequeño flujo de un chatbot para vender autos basado en estados finitos, los números a la izquierda de cada bloque se usan para representar el estado en el que el usuario se encuentra, a la derecha se representa la información que se almacena al final del turno para que en la siguiente interacción el agente sepa qué información extraer y con qué responder.

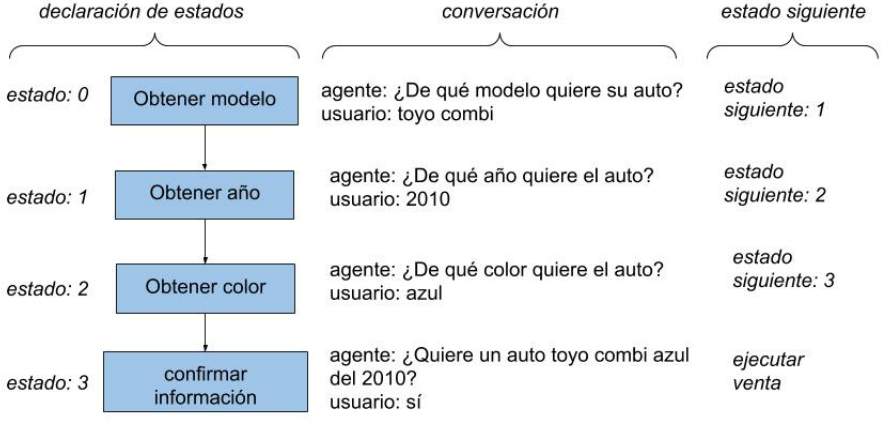

Figura 4.3: Flujo de un agente basado en estados finitos

Cuando escogí esta metodología fue importante almacenar dos ele-

mentos clave en una estructura de datos. El primero es el siguiente estado de la conversación, de esta forma cuando el usuario vuelve a hablar con el agente el sistema sabe qué información debe pedir, el otro elemento está conformado por los datos que se han recabado para cumplir el objetivo, en el caso de la imagen 4.3 son los datos para poder concretar la venta de un automovil. La forma en la que los estados o la información del usuario es almacenada puede variar dependiendo del caso de uso y de la aplicación. Algunas veces lo almacené en memoria usando técnicas de serialización de datos, y en otras ocasiones el equipo de desarrollo de mariachi.io creó bases de datos para poder tener sistemas con mayor escalabilidad.

4.2.2. Modelando agentes basados en *frames*

Como se ha dicho en capítulos anteriores, desarrollar agentes basados en *frames* aporta mayor libertad a la conversación. De forma similar a los agentes de la subsección 4.2.1 también propongo una estructura de datos para poder almacenar la información que se tiene de la conversación y que también funciona para poder generar el diálogo del siguiente turno.

En la figura4.4 podemos ver un *frame* para un agente que vende automóviles, a la izquierda se muestran elementos de un número binario. Éste número es una representación del *frame* si usamos un número entero de $8bits$ podemos representar un *frame* que puede contener hasta ocho elementos, los dígitos con 1 representan un *slot* del que se tiene la información y los que tienen 0 los *slots* con información faltante. Las dos principales ventajas son las siguientes, por un lado tenemos la información sobre los elementos desconocidos para

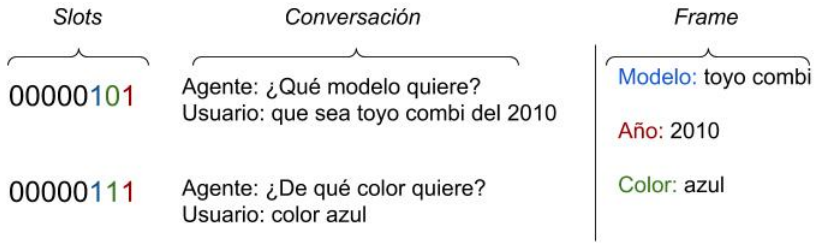

Figura 4.4: Conversación basada en *frames* para la venta de un automovil

completar el objetivo de la charla almacenada en un elemento que utiliza pocos recursos, y por otro lado podemos generar la respuesta recorriendo los digitos de un número entero representado en binario. Algunas otras propuestas sugieren el uso de formatos como *XML* con campos para representar los *slots* [9] sin embargo mi propuesta está enfocada en la optimización de memoria.

4.2.2.1. Bibliotecas de interacción humano chatbot

Los modelos de las secciones 4.2.1 y 4.2.2 sirven como abstracción para poder modelar flujos de la conversación, que aunque en algunas pruebas de concepto las estructuras de datos necesarias fueron programadas por mí, otros miembros del equipo de mariachi.io se encargaron de modelar e implementar las bases de datos necesarias para tener mayor escalabilidad en los agentes. Los elementos más tangibles de mi trabajo en mariachi.io son las bibilotecas de interacción humano chatbot, que sirven para poder implementar el entendimiento de lenguaje natrual necesario en los modelos que propuse en las secciones

4.2.1 y 4.2.2.

Éstas bibliotecas resuelven el problema de encontrar las intenciones y las entidades en el texto del usuario, que tienen la ventaja de dar a mariachi.io mayor control sobre el comportamiento de los algoritmos. De ésta forma mariachi.io puede tener conocimiento que es aplicable en proyectos de diferentes clientes, y que incluso puede ser útil en diferentes tareas de procesamiento de lenguaje natural.

En el siguiente capítulo muestro la propuesta general sobre la metodología que propuse a mariachi.io para poder implementar agentes conversacionales orientados a tareas, basandome en las propuestas de otras compañias. En el siguiente capítulo profundizo sobre la metodología necesaria para replicar las bibliotecas de interacción humano chatbot que desarrollé para mariachi.io.

Característica	*Chatfuel*	*wit.ai*	*IBM watson*	Biliotecas propuestas
Flujo basado en estados finitos	✓		✓	✓
Flujo basado en *frames*				✓
Extracción de intención	✓	✓	✓	✓
Extracción de entidades		✓	✓	✓

Cuadro 4.1: Comparación te características es propuestas de herramientas para desarrollo de agentes basados tanto en metodologías de estados finitos como *frames*.

Capítulo 5

Metodología

En éste capítulo muestro la metodología que utilicé para desarrollar las bibliotecas de interacción humano chatbot que sirven para diseñar agentes basados en estados finitos y *frames*. Como expliqué en el capítulo pasado, las bibliotecas de interacción humano chatbot se usan en distintos proyectos relacionados con agentes converacionales.

Como vemos en la figura 5.1 el objetivo de las bibliotecas es extraer las entidades y predecir la intención de un texto de entrada. En azul y rojo muestro el preprocesamiento y la vectorización respectivamente. Diseñé estas herramientas para limpiar el texto de entrada y dejarlo en un formato que se pueda usar en los algoritmos de clasificación(los elementos en verde).

Los algoritmos de los bloques verdes usan técnicas de aprendizaje automático. En diversas ocasiones desarrollé herramientas de aumentación de datos, con las que generaba más datos que servían como ejemplos para los algoritmos de aprendizaje. Las herramientas de aumentación de datos cambiaron dependiendo del proyecto, pues re-

querían un estudio del problema para poder desarrollarse de forma adecuada.

En color naranja, muestro el proceso de despachar datos. Para éste ejemplo la salida está en formato *JSON*, sin embargo el formato puede cambiar dependiendo de la aplicación.

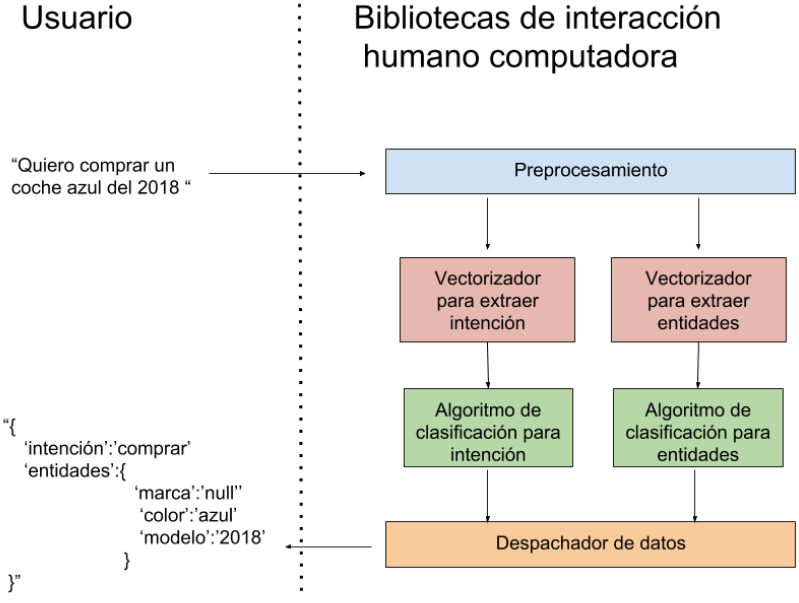

Figura 5.1: flujo de predición de intención y extracción de entidades con bibliotecas de interacción humano computadora

5.1. Preprocesamiento

Usar un buen preprocesamiento es clave para obtener buenos resultados en los algoritmos de detección de intenciones y extracción

de entidades. En mi caso los dos elementos en el preprocesamiento de la entrada del usuario fueron el uso de *stopwords* y un proceso de lematización. Con éstos dos elementos, puedo reducir el espacio de búsqueda en los algoritmos, ésto no solo hace que el funcionamiento sea mejor sino que también optimiza la memoria que se utiliza para la predicción en los algoritmos.

5.1.1. stop words

El lenguaje natural, es el lenguajes que usamos los humanos para comunicarnos, como cualquier lenguaje está compuesto de elementos a los que llamamos palabras. Los *stop words* son palabras que para algunas tareas no son relevantes en el análisis del mensaje como pronombres, preprocisiones o artículos. Cuando removemos los *stop words* logramos reducir el espacio de términos [12]. Algunos stopwords comunes son *a o ahí*, por lo que la oración *voy a la tienda que está ahí* después de someterse a la eliminación de *stop words* queda de la siguiente manera *voy tienda está* .

En mi caso usé la lista de palabras que conforman los *stop words* para el español de el *Natural Language Toolkit* para python [1]. Una vez teniendo la lista, lo que hice fue eliminar del texto de entrada las palabras de la lista.

5.1.2. Lematización

Otra técnica usada para reducir el espacio de búsqueda de los algoritmos, es el proceso de *Lemmatización*. Para definir el proceso

[1] https://www.nltk.org/ en 2018

de Lemmatización es necesario definir las formas flexionadas de las palabras, estas formas son las alteraciones morfológicas de las palabras para poder adaptarse al significado gramatical de la oración por ejemplo plurales, femeninos, masculinos, conjugaciones, etc [10]. En el proceso de lematización se recupera o extrae el lema[2] de una palabra flexionada , por ejemplo en español para las palabras *amaría, amo, amé*, el lema es la palabra *amar*.

Una forma común para modelar el proceso de lemmatización es usar una tabla hash que tiene como llaves un gran número palabras de una lengua que normalmente son las más frecuentes, y como valor los lemas de las palabras, así para cada palabra de la oración del usuario. Guardé la información necesaria para generar la tabla hash en memoria como un archivo json, sin embargo en mi tiempo con mariachi.io hemos explorado usar bases de datos como mongodb para poder tener los lemas en un servidor.

En mi caso la información para generar la tabla hash fue tomada del corpus AnCora[3], que está disponible para español y catalán, el corpus cuenta con diferentes niveles de anotación en mi caso usé el de lemas. Es imporante resaltar que AnCora fue la base de nuestro lematizador, sin embargo conforme el tiempo ha pasado, hemos añadido más palabras como *cheve, cheves, chela* a *cerveza*.

[2]Forma representativa de todas las formas flexionadas de una palabra [10]
[3]http://clic.ub.edu/corpus/es/node/131 en 2018

5.2. Algoritmo Detección de intención

El algoritmo de Detección de intención está basado en técnicas de aprendizaje automático utilizando máquinas de soporte vectorial, por lo que tiene dos etapas principales, la primera es entrenar el algoritmo para detectar intenciones y la segunda es procesar el texto del usuario para predecir la intención del texto de entrada. Como vemos en la imágen 5.2 en la fase de entrenamiento se optimizan los parámetros de diferentes estructuras de datos, que se almacenan en memoria para poder usarse en la fase de predicción.

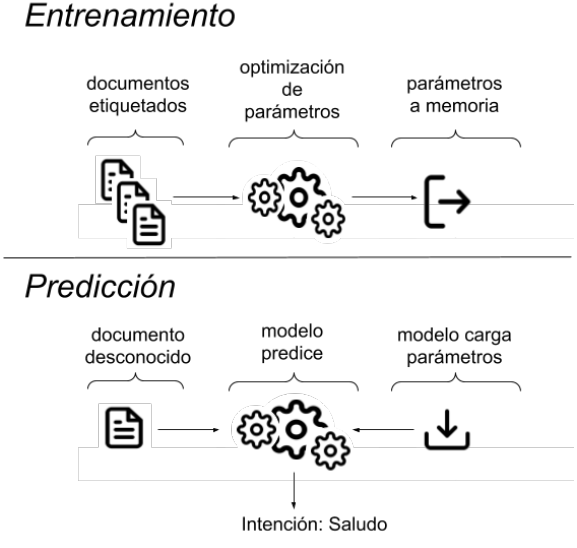

Figura 5.2: flujo de entrenamiento y predicción en los algoritmos de las bibliotecas de interacción humano-chatbot

En mi caso usé la biblioteca pickle[4]. Usar pickle para cargar los

[4]https://docs.python.org/3.6/library/pickle.html en 2018

datos desde memoria es útil en demostraciones o en ambientes de desarrollo, sin embargo para producción el equipo de *backend* de *mariachi.io* ha usado técnicas más avanzadas como las redes de distribución de contenido, o CDN por sus siglas en inglés.

5.2.1. Entrenamiento

El entrenamiento del algoritmo que desarrollé para detectar intenciones toma como entrada texto etiquetado como el que se muestra en la figura 5.3, las salidas de éste proceso son las estructuras de datos optimizadas para hacer la predición de datos. Para el algortimo que desarrollé se necesitan encontrar los parametros de dos elementos, el primero es un vectorizador basado en tf-idf 5.2.1.1 y posteriormente los vectores de soporte 5.2.1.2.

Texto en crudo

texto ,	*intención*
hola cómo estás ,	saludo
cómo te encuentras hoy ,	saludo
me puedes vender un coche,	comprar
quisiera comprar un coche,	comprar
quiero que me vendas un coche,	comprar

Texto preprocesado

texto ,	*intención*
hola como estar ,	saludo
como encuentrar hoy ,	saludo
poder vender coche,	comprar
querer comprar coche,	comprar
querer que vender coche,	comprar

Figura 5.3: datos antes de ser preprocesados, datos después de quitar *stop words* y lemmatizar. Los datos de la derecha son la entrada del proceso de entrenamiento

5.2.1.1. Vectorización con tf-idf

Hay muchas formas para poder representar las oraciones en un espacio vectorial, escoger una representación adecuada depende del problema que se quiere resolver. En mi caso utilicé tf-idf como algoritmo

de vectorización pues ya existía la implementación un la biblioteca scikit-learn [5].

El objetivo de tf-idf es darle un mayor valor a las palabras que se repiten con menos frecuencia en los documentos. Ésto se logra usando dos elementos es la frecuencia de términos o *term frecuence* y la frecuencia inversa de documento o *inverse document frecuence*

Cuando hablamos de términos nos referimos a las palabras que están dentro de un documento que en mi caso son las oraciones preprocesadas del corpues de entrenamiento o las oraciones preprocesadas de la entrada del usuario. De ésta forma hablo de la frecuencia de término me refiero al número de veces que aparece un término en un documento.

La frecuencia inversa de documento se calcula con la siguiente expresión:

$$idf(d,t) = log(\frac{n}{df(d,t)+1})$$

donde n es el número total de documentos y $df(d,t)$ es la frecuencia de documentos es decir el número de documentos que contienen el término t. De ésta forma el valor tf-idf de un documento se calcula de la siguiente manera:

$$\text{tf-idf(d,t)} = tf(t) * log(\frac{n}{df(d,t)+1})$$

Así cuando se vectoriza una oración entera, se calcula el tf-idf de cáda término y ésto genera un vector. Así la oración está lista para ser usada tanto en entrenamiento de un algoritmo o para predecir la intención.

[5] http://scikit-learn.org/stable/ en 2018

Como podemos ver, es necesario calcular *idf* de cáda término, es por eso que en la fase de entrenamiento se calcular éstos valores y se guardan en memoria. De ésta forma en la predicción calcular el tf-idf de una oración es mas fácil y por ende escalable.

5.2.1.2. Máquinas de soporte vectorial

El algoritmo que usé para poder predecir las intenciones de las oraciones fue máquinas de soporte vectorial. Para usar máquinas de soporte vectorial, debemos tener un conjunto de datos $(\overline{x_0}, y_0), ..., (\overline{x_n}, y_n)$ donde $\overline{x_i}$ es un vector de características (oraciones vectorizadas en mi caso), y y_i la categoría a la que pertenece el vector(la intención de la oración). Cuando la clasificación es binaria, y_i puede tomar valores de 1 o -1. En mi caso éstos valores representarían alguna de las intenciones del texto.

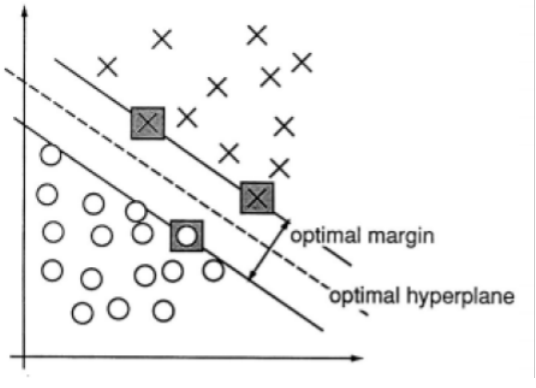

Figura 5.4: hiper plano óptimo, en gris se muestran los vectores de soporte para el hiperplano, imágen tomada de [4]

El objetivo de las máquinas de soporte vectorial es usar hiper-

planos para poder separar vectores de diferentes clases. Como se ve en la imágen 5.4 el hiperplano más optimo es aquel que logra separar las clases teniendo el mayor margen posible. Podemos definir un hiperplano de la siguiente manera:

$$w\overline{x} + b = 0$$

El hiperplano óptimo que se busca obtener con las máquinas de soporte vectorial se puede representar así:

$$\sum \alpha_i z_i x_i + b = 0$$

Dónde α_i es un escalar que multiplica a z_i, un vector de soporte tomado de el conjunto de datos, y $\overline{x}$ es la entrada que el modelo busca clasificar. De ésta forma como la clase y toma valores de 1 o -1, podemos encontrar la clase de un vector $\overline{x}$ de la siguiente manera [4]:

$$y = sgn(\sum \alpha_i z_i x_i + b)$$

Si se quiere hacer una clasificación entre varias clases, existen diferentes técnicas. Éstas técnicas proponen modelar la clasificación multiclase como multiples clasificaciones binarias. Un ejemplo es hacer multiples clasificadores que tomen una clase contra todas las demas, así para cada clase se tiene un clasificador que otorga un valor de certeza y la clase que se predice es la que tenga el mayor valor de certeza.

El proceso de entrenamiento consiste encontrar los vectores de soporte z_i, los escalares α_i, y el intercepto b necesarios para modelar el hiperplano que puede separar las clases dejando el mejor margen. Es

por eso que parte del proceso entrenamiento de el algoritmo incluye almacenar en memoria los vectores de soporte.

En mi caso utilicé la bibloteca scikit-learn [6] de google para poder implementar tanto máquinas de soporte vectorial como el vectorizador tf-idf. Para serializar scikit-learn cuenta con un módulo llamado joblib que está diseñado para almacenar los datos de sus clasificadores y vectorizadores.

5.2.2. Predicción de intención

En la parte de predicción el algoritmo busca poder decir cuál es la intención de la oración de entrada. Éste proceso usa los elementos que se serializaron en la parte del entrenamiento del algoritmo. Como podemos ver en la imágen 5.5, el proceso de predicción es mas simple que el proceso de entrenamiento. Lo que se busca es cargar en memoria tanto el vectorizador como el clasificador, para que de ésta forma se vectorice el texto que en el que el usuario se comunica con el chatbot y posteriormente se clasifique el vector. Así en la biblioteca de interacción humano-computadora la función de predicción de intención regresa la intención de la oración.

5.3. Algoritmo Extracción de entidades

El algoritmo de extracción de entidades busca poder regresar los elementos de interés o que son necesarios para poder cumplir los objetivos de la tarea para la que se diseña el agente conversacional. Como

[6] http://scikit-learn.org/stable/ en 2018

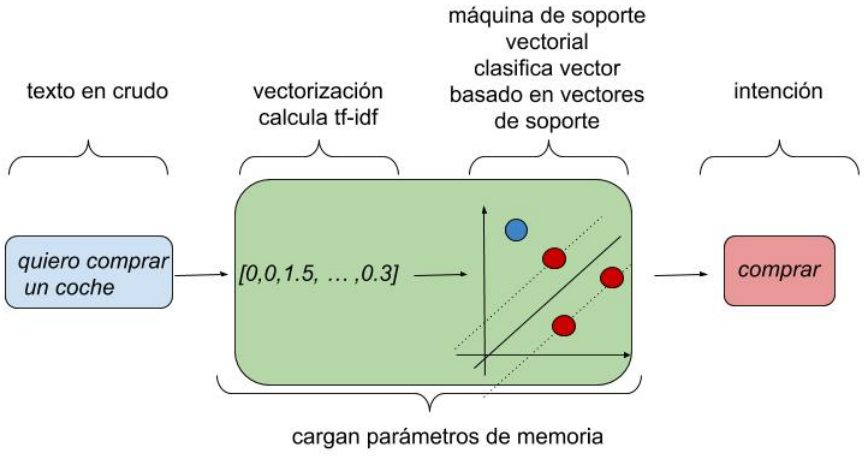

Figura 5.5: proceso para saber la intención de un texto del usuario. En azul muestro la entrada, en rojo la salida y en verde el proceso interno de la biblioteca.

se ve en la imágen (referenciar imágen), el algoritmo asigna a cada palabra una clase, que puede referenciar a una entidad o un valor nulo representado por "*".

5.3.1. Entrenamiento

Al igual que en el algoritmo anterior, el entrenamiento busca optimizar los parámetros de una máquina de soporte vectorial que puede predecir para cada palabra cuál es la entidad que la representa. Sin embargo la diferencia radica en que no se quiere clasificar toda la oración sino parte de la palabra, es por eso que utilicé una forma distinca de vectorizar los elementos de la oración.

5.3.1.1. Vectorización con n-gramas

Al igual que en el algoritmo de detección de entidades, necesitamos una herrmaienta que nos ayude para poder convertir el texto en vectores. La diferencia es que ahora buscamos representar palabras y no oraciones. Es por ésto que propuse el uso de n-gramas.

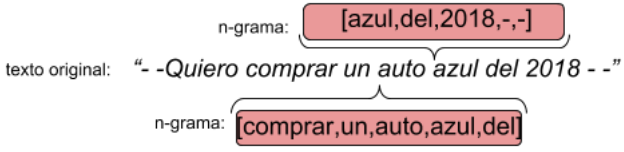

Figura 5.6: n-gramas de una oración

En general, podemos decir que un n-grama es una secuencia de n elementos o gramas. En mi caso uso n-gramas de palabras. En la figura 5.6 vemos dos n-grama de cinco elementos, también podemos ver la transformación a índices. Los indices se generan mapeando todo el vocabulario con el que se entrena el vectorizador, de ésta forma cáda palabra tiene un índice.

De ésta forma, lo que se busqué es tener una representación donde la palabra central en el n-grama es la palabra que deseaba clasificar. Sin embargo usé también información de las palabras alrededor para poder tener información del contexto de la entidad. Es por eso que en la figura 5.6 el segundo n-grama tiene guiones a la izquierda, de ésta forma podemos representar la primera palabra e una oración.

Dado que hay que guardar los índices, para poder mapear las palabras de los n-gramas, éste vectorizador también necesita ser almacenado en memoria para usarse en la fase predicción. Para implementar

ésta vectorización, utilicé la biblioteca "DictVectoricer"de sklearn [7], ésta biblioteca permite vectorizar diccionarios, es por ésto que utilice diccionarios para poder representar n-gramas. Con ésta forma de representar los n-gramas se toma únicamente en cuenta la aparición de los términos en el n-grama.

Al igual que en el algoritmo de detección de entidades, el algoritmo que usé para clasificar fue máquinas de soporte vectorial. En éste caso el vector de entrada fue distinto, pero la implementación del clasificador fue la misma.

5.3.2. Predicción de entidades

Para poder predecir las palabras es necesario poder generar los n-gramas vectorizados. Por lo que los pasos para predecir son, al texto de entrada, concatenar dos guiones al inicio y al final. Posteriormente se recorre la oración para generar los n-gramas de cada palabra. La máquina de soporte vectorial predice las clases de cada n-grama.

Una vez que se tienen las predicciones en una lista, y la oración del usuario en otra, lo que se busca es regresar la información en un formato que sea fácil de consumir, mi propuesta fue usar el formato json como formato para la salida de los elementos del extractor de entidades. En la figura 6.1 podemos ver el flujo de una oración la que se le extraen las entidades.

[7] http://scikit-learn.org en 2018

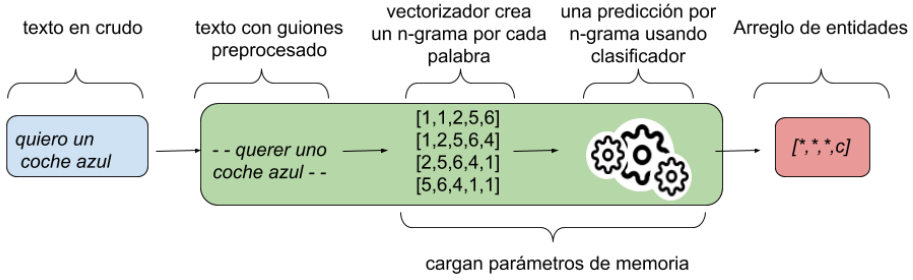

Figura 5.7: extracción de entidades de una oración. En azul muestro la entrada, en rojo la salida y en verde los procesos internos de la biblioteca.

5.4. Selección de configuración del modelo

Cuando se utiliza un modelo de aprendizaje automático, es necesario tomar en cuenta dos elementos importantes: la selección de los parámetros del modelo, y las métricas de evaluación que dan claridad sobre el desempeño del algoritmo.

5.4.1. Parámetros en máquinas de soporte vectorial.

El principal problema al utilizar clasificadores lineales, es que en algunas ocaciones los datos no pueden ser separados por una recta. Para mitigar este problema los algoritmos permiten a los usuarios configurar una serie de parámetros. En ésta sección muestro cual fue la forma en la que seleccioné los parámetros de las máquinas de soporte vectorial tanto para la extracción de entidades como en la clasificación

de intenciones.

Los parámetros que se utilizan para configurar a las máquinas de soporte vectorial, que se utilizan normalmente son: el kernel de la máquina de soporte vectorial, el parámetro de regularización C y cuando el kernel es no lineal el parámetro gamma σ.

El truco de kernel, consiste en utilizar transformaciones que pueden ser basados en ecuaciones lineales, polinómicas, curvas gaussianas entre otras [5]. En mi caso siempre utilicé un kernel lineal, por lo que hablar sobre el parámetro gamma no es necesario.

El otro parámetro relevante es C, que sirve para aumentar el márgen entre vectores de soporte cercanos la recta del hiperplano. De ésta forma algunas características de los ejemplos en el entrenamiento pueden ser ignorados por el algoritmo. En la figura (referenciar figura) se muestra un ejemplo de varios valores del parámetro C en un conjunto de datos. Entre mas grande es el valor de C más características se ignoran

En la figura 5.8 podemos ver varias configuraciones del parámetro C. Como podemos ver, entre mayor es el valor de C se toman menos características para el entrenamiento. Por el contrario cuando el valor de C es pequeño, se toman en cuenta un número superior de vectores de soporte.

No es fácil escoger el correcto valor del parámetro C [1]. En mi caso el valor fue cambiando dependiendo del proyecto. La forma de escoger el parámetro fue utilizando métricas de evaluación.

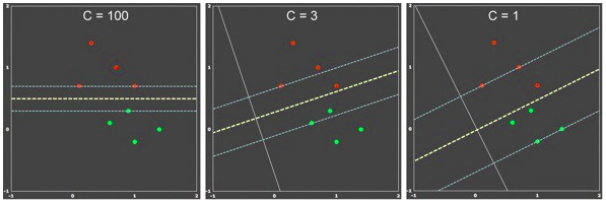

Figura 5.8: Distintos hiperplanos variando el parámetro C [8]

5.4.2. Métricas de Evaluación

Las métricas de evaluación nos ayudan para plasmar el desempeño de nuestro modelo en una forma cuantitiva. El objetivo de éstas medidas se basa en contabilizar de las prediciones realizadas cuales están equivocadas y cuales son correctas.

		Certeza	
		Certeza Positiva	Certeza Negativa
Predicciones	Predicción Positiva	Verdadero Positivo (VP)	Falso Positivo (FP)
	Predicción Negativa	Falso Negativo (FN)	Verdadero Negativo (VN)

Figura 5.9: Comparación de certeza y predicción

Supongamos que tenemos un conjunto de datos etiquetados con únamente dos clases una positiva y una negativa. Con éste conjunto deseamos saber cuál es el rendimiento de nuestro algoritmo. En la figura 5.9 podemos ver la comparación entre las prediciones que hace un modelo y la certeza o las etiquetas del conjunto de datos que separamos.

De ésta forma los valores *verdadero positivo (VP) y verdadero negativo falsos (VN)* son las predicciones correctas, mientras que las *falso positivo(FP) y falso negativo(NF) son las incorrectas*. Cuando se toma en cuenta la ocurrencia de éstos elementos se pueden tener dos metricas de evaluación: *recall* y *precisión*.

Precisión abstrae proporcion de prediciones correctas*(VP)* de una clase con respecto al total de veces que se predijo dicha clase*(VP + FP)*. Por lo que la forma de medir la precisión es:

$$precisión = \frac{VP}{VP+FP}$$

De otra forma *recall* puede cuantificar la relación entre las predicciones positivas correctas*(VP)* y el total de muestras que se sabe son positivas*(VP + FN)*.

$$recall = \frac{VP}{VP+FN}$$

La medida que utilicé para poder saber el desempeño de mis modelos fue $F1score$. Que resulta como media armónica de las dos medidas anteriores.

$$F1score = 2\frac{presicion \cdot recall}{presicion + recall}$$

De ésta forma para cada algoritmo pude entender cuál era el desempeño con una sola métrica. Éste valor me ayudó a configurar los parámetros de una máquina de soporte vectorial, identificar cuando hay sobre ajuste o cuando es muy dificil para el modelo separa las clases.

Sin embargo, dado que las tareas relacionadas con la selección de datos de entrenamiento como etiquetar, o aumentar los datos las hacía

yo. Muchas veces fue mas valioso probar el algoritmo con humanos y probar el desempeño de una forma más cualitativa.

En éste capítulo mostré la metodología que utilicé para poder extraer entidades en la bibliotecas de interacción humano chatbot que desarrollé par ala empresa mariachi.io. Con éstas bibliotecas es posible desarrollar agentes conversaciones para las metodologías basadas en agentes finitos y basadas en *frames*.

Capítulo 6

Resultados

En éste capítulo muestro los resultados que han tenido las bibliotecas de interacción humano chatbot, que desarrollé para la empresa *mariachi.io*. Muestro los resultados en proyectos que la empresa mariachi.io generó, en los cuales se usaron las bibliotecas o las metodologías que realicé para la empresa.

Los primeros casos en los que aplicamos las metodologías basadas en estados finitos y basadas en *frames* fueron en hackathones. En éstos concursos generamos agentes conversacionales con diferentes funciones. El más importante de éstos sistemas fue *chepoventa* que fue presentado en el Hackathon de BBVA bancomer en octubre del 2016.

El objetivo de éste sistema era encontrar elementos en la tienda de *mercado libre*, usamos las bibliotecas para extraer los elementos necesarios del texto, sin embargo la mayor inovación fue que usé un sistema de visión por computadora que podía extraer los nombres de los objetos de la escena. Éste chatbot ganó el premio principal de la competencia y dos premios secundarios.

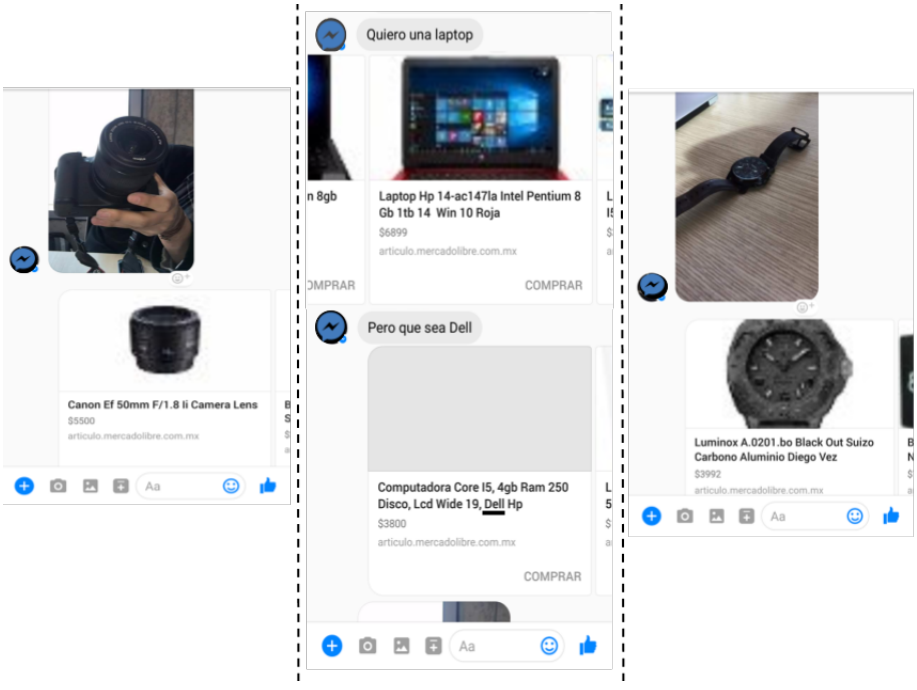

Figura 6.1: Ejemplos de conversaciones con el chatbot *chepoventa*

Como firmé *acuerdos de no divulgación* no puedo explicar a detalle el funionamiento de varios sistemas en los que se usaron las bibliotecas de interacción humano computadora. Sin embargo sí puedo decir que las bibliotecas se han usado como principal motor de procesamiento de lenguaje natural en distintos proyectos. También han servido como complemento en sistemas más grandes que usan distintos motores para procesaro el lenguaje natural. Éste es el caso de *Roomie Bot* que es un robot con diferentes funcionalidades como buscar contenido en wikipedia, y controlar dispositivos del horgar.

Las bibliotecas de interacción humano-chatbot son ahora un ele-

mento con el que mariachi puede utilizar las metodologías basadas en estados finitos y basadas en *frames*. Sin embargo las propias metodologías son herramientas que sirven para desarrollar agentes de forma más rápida o incluso generar sistemas más complejos.

Actualmente *mariachi.io* está generando un producto propio, un generador de agentes conversacionales llamado *Tok Tok*. Estos agente busca poder generar ventas de alimentos para restaurantes, y usan las metodologías basadas en estados finitos y basadas en *frames* para modelar las conversaciones.

El producto sigue en desarrollo, por lo que aún no implementan las bibliotecas de interacción humano-chatbot. Sin embargo su implementación ya está planeada.

El mejor resultado que tienen las bibliotecas de interacción humano computadora que desarrollé para la empresa *mariachi.io* está en la portabiliad que tienen para adaptarse en multiples proyectos de ingeniería tanto ahora, como en el futuro. Lo que hace que las bibliotecas sean un elemento útil para la empresa *mariachi.io*.

Capítulo 7

Conclusiones

En éste capítulo presento una recapitulación del trabajo que realicé para la empresa *mariachi.io*. Que son las bibliotecas de interacción humano chatbot. Éstas bibliotecas sirven para desarrollar sistemas de diálogo, basándose en metodologías basadas en agentes finitos y *frames*. Además hago propuestas para trabajo futuro de las bibliotecas de interacción humano-chatbot basadas en mi experiencia profesional.

Por el lado de las metodologías basados en estados finitos y basados en *frames*, el equipo de desarrolladores de *mariachi.io* ha logrado generar un buen avance. Los objetivos ahora se enfocan en crear mejores bases de datos para almacenar la información de los usuarios.

Las bibliotecas propuestas han sido utilizadas en dos agentes de mediana escala *chepo venta*, *Roomie bot*, una de gran escala *Dinero Express*, se utilizará en un agente de gran escala que es *Tok Tok*. Además es parte del conocimiento de la empresa *mariachi.io* por lo que se utilizará en mas proyectos.

En las bibliotecas de interacción humano chatbot, aunque han so-

lucionado muchos problemas y forman parte principal del desarrollo de agentes conversacionales. Requieren una mejor optimización, mi trabajo actual se enfoca en crear mejores clasificadores.

He propuesto nuevas formas de clasificar texto basadas en redes bayesianas, que prometen generar mejores resultados. Otro tema importante es mejorar la aumentación de datos, hemos probado experimentos que involucran usuarios reales o *screapers* que nos puedan dar más información.

Un punto a nuestro favor, es que *mariachi.io* ha empezado a crecer, y tiene clientes potenciales que cuentan con una mayor cantidad de datos. Ésto hace que el producto final pueda ser mejor, y que podamos empezar a usar clasificadores más complejos, como los basados en redes neuronales.

Ahora, como *mariachi.io* tiene herramientas que facilitan el desarrollo de agentes conversacionales, he podido asumir tareas en proyectos distintos. Algunas de éstas tareas distintas involucran temas de visión por computadora o incluso invertir tiempo en generar una comunidad de desarrolladores.

En conclusión, durante el tiempo que he trabajo para *mariachi.io*, he desarrollado bibliotecas lo suficientemente eficientes para resolver los problemas de consultoría. En éstos problemas he podido adaptar la solucion a las necesidades y recursos de los clientes de la empresa obteniendo resultados satisfactorios.

El nuevo camino es el de la mejora de las herramientas, que supone un mayor crecimiento tanto para *mariachi.io* como para mi en mi desarrollo profesional. Pues se tienen mas recursos, pero también más responsabilidades.

Bibliografía

[1]

[2] Suket Arora, Kamaljeet Batra, and Sarabjit Singh. Dialogue system: A brief review. *CoRR*, abs/1306.4134, 2013.

[3] R Carpenter. Cleverbot [computer program], 2015.

[4] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[5] Martin Hofmann. Support vector machines—kernels and the kernel trick. *Notes*, 26, 2006.

[6] Jason L Hutchens and Michael D Alder. Introducing megahal. In *Proceedings of the Joint Conferences on New Methods in Language Processing and Computational Natural Language Learning*, pages 271–274. Association for Computational Linguistics, 1998.

[7] Nobuo Inui, Takuya Koiso, Junpei Nakamura, and Yoshiyuki Kotani. Fully corpus-based natural language dialogue system. In *Natural Language Generation in Spoken and Written Dialogue, AAAI Spring Symposium*, 2003.

[8] Dan Jurafsky and James H Martin. *Speech and language processing*, volume 3. Pearson London:, 2014.

[9] William K Thompson and Harry M Bliss. Frame goals for dialog system, February 2 2010. US Patent 7,657,434.

[10] Elizabeth Luna Traill, Alejandra Vigueras Ávila, and Gloria Estela Baez Pinal. *Diccionario básico de lingüística*. UNAM, 2005.

[11] C Vásquez. Una aproximación a la psicoterapia de carl rogers. *Actualidad Psicológica*, 4(1), 2012.

[12] S Vijayarani, Ms J Ilamathi, and Ms Nithya. Preprocessing techniques for text mining-an overview. *International Journal of Computer Science & Communication Networks*, 5(1):7–16, 2015.

[13] Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45, January 1966.