

Centro de Formación permanente

Universidad de Sevilla

Sevilla, 2017

Autor: Inmaculada Perea Fernández

Tutor: Fernando Sancho Caparrini

**Asistente conversacional para frigorífico inteligente**

Trabajo Fin de Máster

Data Science y Big Data

Trabajo Fin de Máster

Data Science y Big Data

**Asistente conversacional para frigorífico inteligente**

Autor:

Inmaculada Perea Fernandez

Tutor:

Fernando Sancho Caparrini

Centro de Formación Permanente

Universidad de Sevilla

Sevilla, 2017

Proyecto Fin de Máster: Asistente conversacional para frigorífico inteligente

|  |  |
| --- | --- |
| Autor: | Inmaculada Perea Fernández |
| Tutor: | Fernando Sancho Caparrini |

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2017

El Secretario del Tribunal

*A mi familia*

*A mis maestros*

**Agradecimientos**

Texto

*Inmaculada Perea Fernández*

*Sevilla, 2017*

**Resumen**

Texto

**Abstract**

Texto

Indice

[Agradecimientos ix](#_Toc496709606)

[Resumen xi](#_Toc496709607)

[Abstract xiii](#_Toc496709608)

[Índice de Tablas xvii](#_Toc496709609)

[Índice de Figuras xviii](#_Toc496709610)

[Notación xix](#_Toc496709611)

[1 Introducción 11](#_Toc496709612)

[1.1 Motivación 11](#_Toc496709613)

[1.2 Objetivos 11](#_Toc496709614)

[1.3 Definición de requisitos 11](#_Toc496709615)

[1.3.1 Requisitos funcionales 11](#_Toc496709616)

[1.4 Organización de la memoria 12](#_Toc496709617)

[2 Estado del arte 10](#_Toc496709618)

[2.1 Frameworks para chatbots 11](#_Toc496709619)

[2.2 APIs NLP 11](#_Toc496709620)

[2.2.1 Conversation (IBM Watson) 12](#_Toc496709621)

[2.2.2 Lex (Amazon) 13](#_Toc496709622)

[2.2.3 Api.ai (Google) 14](#_Toc496709623)

[2.2.4 Luis (Microsoft) 15](#_Toc496709624)

[2.2.5 Wit.ai (Facebook) 15](#_Toc496709625)

[2.2.6 SiriKit (Apple) 16](#_Toc496709626)

[2.2.7 Criterios de selección 17](#_Toc496709627)

[2.2.8 Conclusiones 23](#_Toc496709628)

[2.3 Introducción a servicios IBM Watson 24](#_Toc496709629)

[2.3.1 Conversation 24](#_Toc496709630)

[2.3.2 Speech to Text 24](#_Toc496709631)

[2.3.3 Text to Speech 24](#_Toc496709632)

[2.3.4 Visual recognition 24](#_Toc496709633)

[3 Trabajo realizado 25](#_Toc496709634)

[3.1 Arquitectura de la solución 25](#_Toc496709635)

[3.2 Interfaz de usuario 27](#_Toc496709636)

[3.2.1 Selección de plataforma 27](#_Toc496709637)

[3.2.2 Integración con Slack 28](#_Toc496709638)

[3.3 Integración con base de datos 29](#_Toc496709639)

[3.3.1 Conexión 29](#_Toc496709640)

[3.3.2 Modelo de datos 29](#_Toc496709641)

[3.4 Configuración y entrenamiento del servicio Conversation 29](#_Toc496709642)

[3.4.1 Intenciones 29](#_Toc496709643)

[3.4.2 Entidades 29](#_Toc496709644)

[3.4.3 Variables de sistema 29](#_Toc496709645)

[3.4.4 Variables de contexto 29](#_Toc496709646)

[3.4.5 Flujo de Diálogo 29](#_Toc496709647)

[3.5 Configuración y entrenamiento del servicio Visual recognition 29](#_Toc496709648)

[3.6 Implementación de la aplicación orquestadora 29](#_Toc496709649)

[3.7 API de recetas de cocina (Food2Fork) 29](#_Toc496709650)

[3.7.1 Selección de la API 29](#_Toc496709651)

[3.7.2 Integración 30](#_Toc496709652)

[3.8 Adaptación para nuevo caso de uso: asistente para tienda de ropa 30](#_Toc496709653)

[4 Conclusiones 31](#_Toc496709654)

[4.1 Resumen del trabajo realizado y cumplimiento de objetivos 31](#_Toc496709655)

[4.2 Dificultades, aportaciones y contribuciones 31](#_Toc496709656)

[4.3 Resultados: importancia, repercusión y utilidad 31](#_Toc496709657)

[5 Líneas futuras 32](#_Toc496709658)

[5.1 Comunicación de voz en tiempo real (streaming) 32](#_Toc496709659)

[5.2 Integración con Voice Gateway 32](#_Toc496709660)

[5.3 Integración con supermercados online 32](#_Toc496709661)

[5.4 Integración con redes sociales 32](#_Toc496709662)

[5.5 Funcionalidad lista de la compra 32](#_Toc496709663)

[5.6 Agenda personal 32](#_Toc496709664)

[5.7 Multi-idioma 32](#_Toc496709665)

[5.8 Coach nutricional 32](#_Toc496709666)

[5.9 Análisis de sentimientos 32](#_Toc496709667)

[5.10 Integración con IBM IoT 32](#_Toc496709668)

[5.11 Notificaciones productos perecederos 32](#_Toc496709669)

[5.12 Cuadro de mando 32](#_Toc496709670)

[Apéndice I: código 33](#_Toc496709671)

[Apéndice II: Guía de instalación 34](#_Toc496709672)

[Apéndice III: Manual de usuario 35](#_Toc496709673)

[Referencias 36](#_Toc496709674)

[Índice de Conceptos 38](#_Toc496709675)

[Glosario 40](#_Toc496709676)

**Índice de Tablas**

**No se encuentran elementos de tabla de ilustraciones.**

**Índice de Figuras**

Figura 1: Ecosistema de chatbots 10

Figura 2: Evolución temporal principales APIs de NLP 12

Figura 3 Arquitectura de la solución 21

Figura 4 Interfaz usuario: Slack 22

**Notación**

|  |  |
| --- | --- |
| A\* | Conjugado |
| c.t.p. | En casi todos los puntos |
| c.q.d. | Como queríamos demostrar |
| ∎ | Como queríamos demostrar |
| e.o.c. | En cualquier otro caso |
| E | número e |
| IRe | Parte real |
| IIm | Parte imaginaria |
| Sen | Función seno |
| Tg | Función tangente |
| Arctg | Función arco tangente |
| Sen | Función seno |
| sin*xy* | Función seno de *x* elevado a *y* |
| cos*xy* | Función coseno de *x* elevado a *y* |
| Sa | Función sampling |
| Sgn | Función signo |
| Rect | Función rectángulo |
| Sinc | Función sinc |
| ∂y ∂x  *x*◦ | Derivada parcial de *y* respecto  Notación de grado, *x* grados. |
| Pr(*A*) | Probabilidad del suceso *A* |
| SNR | Signal-to-noise ratio |
| MSE | Minimum square error |
| : | Tal que |
| < | Menor o igual |
| > | Mayor o igual |
| \ | Backslash |
| ⇔ | Si y sólo si |

# Introducción

## Motivación

Resumen de la situación inicial del problema a resolver.

Justificar la necesidad/interés de abordar ese problema con técnicas de Data Science (incluir ejemplo si se cree conveniente).

Aplicaciones como Whatsapp, Facebook Messenger o WeChat, entre otras, está acaparando la conversación en Internet.

Una interfaz de usuario conversacional nos permite interactuar con una aplicación de forma similar a como interactuamos con otros humanos (lenguaje natural). La intención se puede expresar de diferentes maneras.

## Objetivos

Con este trabajo se quiere obtener lo siguiente:

* Realizar un análisis crítico de algunos de los servicios IBM Watson: Conversation, Visual Recognition, Speech to Text, Text to Speech, Tone analyzer, Natural Language Understanding, Knowledge Studio.
* Comparativa con otros productos del mercado.
* Implementación de un asistente conversacional para frigorífico inteligente.
* Evaluación de los resultados: inconvenientes y ventajas encontrados durante la ejecución del trabajo.

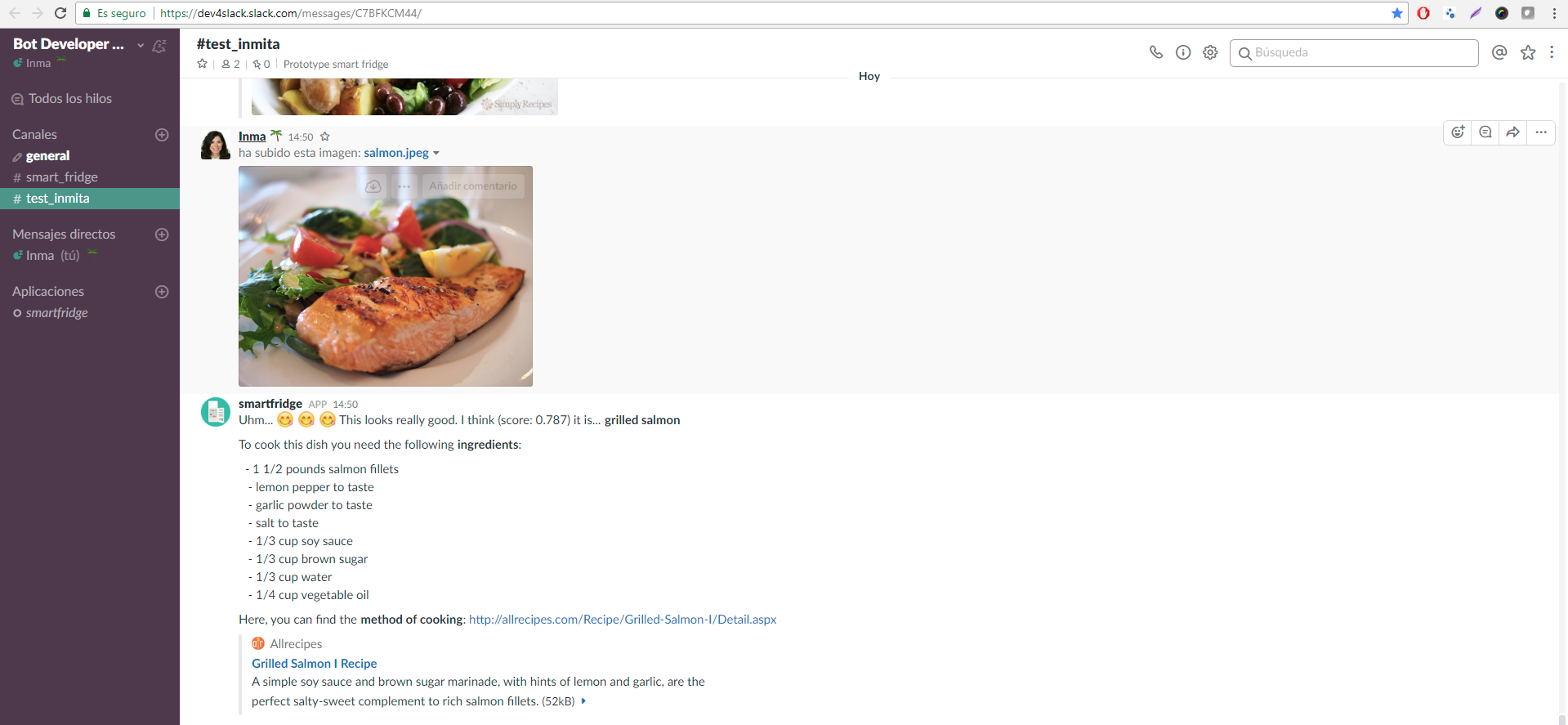
## Definición de requisitos

Es necesario que la conversación sea abierta, es decir, permitiremos que el cliente escriba texto libre en todo momento, pero partiremos de una alcance inicialmente acotado.

A continuación los requisitos funcionales.

### Requisitos funcionales

* [RF01] El asistente estará escuchando y saludará al usuario cuando éste se dirija a él por primera vez.
* [RF02] El asistente será capaz de entender la intención del usuario cuando este se dirija a él usando lenguaje natural, y proporcionar la respuesta más adecuada.
* [RF03] El asistente podrá sugerir platos que cocinar con algunos de los ingredientes con los que cuenta en ese momento en la nevera y con otros posibles ingredientes de los que no disponga en ese momento. Los platos devueltos son los que tengan una mayor puntuación o sean más populares.
* [RF04] El asistente podrá sugerir platos que cocinar exclusimamente con los ingredientes con los que cuenta en la nevera, sin necesidad de comprar ningún ingrediente adicional.
* [RF05] El asistente podrá sugerir platos que cocinar según el tipo de comida que sugiera el usuario: española, francesa, china, italiana, vegana, baja en calorías, especiada, postres, rápida, sin gluten, sin lactosa…
* [RF06] El asistente avisará proactivamente al usuario si alguno de los alimentos que contiene en la nevera va a caducar en menos de 2 días. Esta acción la realiza consultando la fecha de caducidad asociada a los alimentos con los que cuenta en su base de datos.
* [RF07] Si el usario comparte una foto o un enlace de un plato que le apetece comer, el asistente clasificará e identificará de qué plato se trata, y proporcionará los ingredientes del mismo, así como un enlace a la receta.



* [RF08] Tras identificar el plato del que se trata a partir de la foto y extraer los ingredientes necesarios para cocinar dicho plato, el asistente proporcionará al usuario una lista de los ingredientes con los que no cuenta en el frigorífico o la despensa, y que por tanto tendría que comprar.
* [RF09] El frigorífico le avisa si queda poca cantidad de algo que tenga dentro para que lo reponga. Esto lo podría hacer añadiendo una especie de contador a cada elemento de la tabla de productos del frigorífico y por cada vez que el usuario saque de la nevera ese producto se va descontando, ahora mismo tengo un atributo en el modelo de datos que es la cantidad en gramos.
* [RF10] Salvar recetas favoritas del usuario cuando le diga que le ha gustado mucho. Y que luego se las sugiera cuando le diga que tiene ganas de cocinar, o que tiene invitados…

## Organización de la memoria

# Estado del arte

A

ctualmente existe una increíble cantidad de plataformas y herramientas para la creación de chatbots, con diferentes niveles de complejidad, capacidad expresiva y capacidad de integración.

A continuación veremos las herramientas, frameworks e implementaciones más extendidas en la actualidad, y seleccionaremos la que mejor se ajuste a nuestras necesidades.

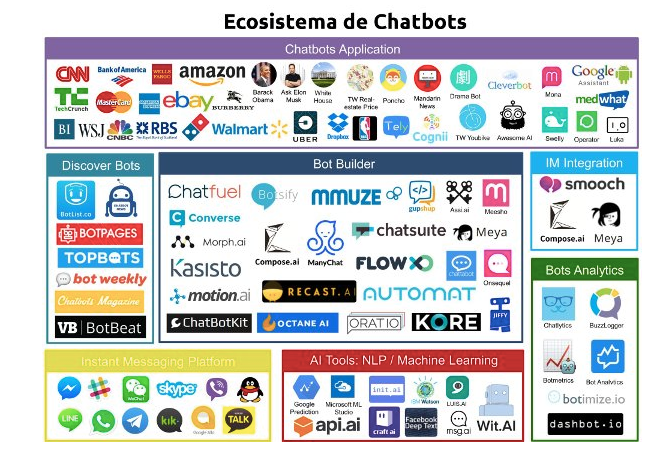


Figura : Ecosistema de chatbots

Según el caso de uso abordado por el chatbot, algunas plataformas son más apropiadas que otras. La selección de la plataforma dependerá del tipo de chatbot que se desee construir, y cuál será su cometido. Si debe tener fuertes habilidades de conversación o por el contrario está orientado a un objetivo específico, y no necesita entender profundamente lo que el usuario dice o recordar todo el contexto de la conversación (chatbots de entretenimiento o chatbots para FAQs)

La mayoría de aplicaciones de mensajería instantánea han incorporado facilidades para el lanzamiento de chatbots en sus plataformas, ya sean para la distribución de contenidos, la atención al cliente, el comercio electrónico, etc. En torno a ellas han aparecido multitud de startups que están haciendo negocio ofreciendo servicios para la creación de bots en pocos pasos y sin necesidad de programar. Son una puerta abierta para los usuarios sin conocimientos técnicos, aunque por ahora tienen muchas limitaciones y se utilizan como un campo de exploración más que de negocio.

Construir un asistente robusto de inteligencia artificial todavía es bastante complicado. Los gigantes tecnológicos abordan este problema proporcionando soluciones de comprensión de lenguaje natural (NLU) que los desarrolladores podemos usar para aumentar nuestras aplicaciones con capacidades de lenguaje natural.

Podemos distinguir por tanto 2 tipos de plataformas o estrategias para la construcción de chatbots:

* Frameworks para chatbots
* APIs NLP

## Frameworks para chatbots

Son plataformas orientadas a usuarios no técnicos. Por lo general, es fácil codificar un chatbot sin tener habilidades de programación y sin tener conocimientos de aprendizaje automático o de procesamiento de lenguaje natural. La idea clave es que el **usuario no tenga que preocuparse por los detalles técnicos**, y poder lanzar bots en tres o cuatro pasos en plataformas de mensajería o redes sociales como Facebook Messenger, Telegram, Slack o WeChat.

Hay una **gran cantidad de frameworks para chatbots**, estos son algunos de los más extendidos:

* [**Chatfuel**](https://chatfuel.com/): uno de los más usados del mercado, fue lanzado en 2015 por los desarrolladores rusos [Dmitry Dumik](https://www.linkedin.com/in/ddumik) y [Artem Ptashnik](https://www.linkedin.com/in/artem-ptashnik-828163bb). Por ahora estos chatbots están disponibles para Facebook Messenger y Telegram. Existen chatbots implementados con este framework en los portales de noticias [Forbes](https://es.wikipedia.org/wiki/Forbes) y [TechCrunch](https://es.wikipedia.org/wiki/TechCrunch" \o "TechCrunch).
* [**ManyChat**](https://manychat.com/): Sirven para invitar a los usuarios a suscribirse al sitio a través de Facebook Messenger y tener una comunicación interactiva con ellos. Existe una versión gratuita y otra de pago. Proporciona una interfaz drag and drop.
* [**Massively**](http://www.massively.ai/): plataforma de desarrollo de bot de código abierto, se pueden implementar en cualquier plataforma de chat (Kik, Facebook Messenger, Skype, Telegram, etc.) o incrustados en un sitio web. El servicio de desarrollo es gratuito, pero la compañía también ofrece un servicio de soporte a los clientes.
* [**Motion.ai**](https://www.motion.ai/): Dispone de una interfaz gráfica para crear flujos de conversación. Proporciona una API para integración con otras aplicaciones y APIs de NLP poco potentes. También incorpora un cuadro de mando por defecto para visualizar los indicadores más relevantes de las conversaciones del bot.
* **Recast.ai:** plataforma colaborativa para construir, entrenar, desplegar y monitorizar bots. Es mas que un framework para bots, porque dispone además de una potente API que proporciona funciones de procesamiento de texto, conversación (extracción de intención, entidades) y análisis de imágenes. Es gratuita para desarrolladores siempre que el código sea público en Github, pero también tiene tarifas para proyectos privados. Soporta multiples idiomas (inglés, francés y anuncian que próximamente en español).

Aunque todas estos framework parezcen muy similares, existen importantes diferencias en la madurez, la usabilidad de la GUI y el poder de procesamiento del lenguaje natural.

Las ventajas que presentan este tipo de plataformas es que tienen una **baja curva de aprendizaje** y por tanto se puede desarrollar un chatbot muy rápidamente.

Suelen proporcionar una **interfaz gráfica** de usuario, a veces esta GUI no son muy intuitivas, y cuando la lógica del chatbot se vuelve más compleja, se vuelve difícil de manejar.

Todas ellas están orientadas a realizar tareas, el ejemplo más común es "pedir una pizza". Son ideales para bots simples, pero tienen **poca o ninguna capacidad de procesamiento del lenguaje natural**. Algunas de estas plataformas no pueden realizar extracción de información.

Este tipo de plataformas **no son válidas para proyectos comerciales a gran escala**. Las conversaciones no pueden ser muy complejas y, por lo general, no es posible integrar recursos externos, como componentes específicos de PNL y ML.

Sin embargo, son plataformas realmente buenas para proyectos a pequeña escala, por lo general para agregar rápidamente una funcionalidad de chatbot a una página, como por ejemplo Chatfuel en Facebook.

## APIs NLP

El **lenguaje natural** es un elemento **fundamental** para la construcción de **chatbots inteligentes**, con los que el usuario pueda interaccionar libremente sin necesidad de seguir un conjunto de reglas básicas. Y que a la vez sean capaces de proporcionar un rango amplio de posibles respuestas.

Desde un punto de vista conceptual, hay dos técnicas principales de programación del lenguaje natural que se han hecho populares con las tecnologías chatbot:

* **Procesamiento del lenguaje natural** (NLP): abarca varias disciplinas que abordan la interacción entre los sistemas informáticos y los lenguajes naturales humanos. Incluye varias subdisciplinas como el análisis del discurso, la extracción de relaciones, la comprensión del lenguaje natural y algunas otras áreas de análisis del lenguaje.
* **Comprensión del lenguaje natural** (NLU): es un subconjunto de NLP que se centra en la comprensión lectora y el análisis semántico.

El software NLP no busca palabras clave en el texto como un motor de búsqueda, sino que utiliza el conocimiento de la **estructura de oraciones, expresiones idiomáticas y patrones aprendidos** para intentar hacer coincidir lo que dice el usuario con una **intención** que ha sido clasificada. Esto se traduce en que el chatbot debe ser programado para identificar ciertas cosas que la gente quiere de él, y **actuar sobre ellas**. Cada vez que una intención se clasifica y se usa en una conversación, el chabot puede proporcionar una acción o respuesta rápida. La **inteligencia** del chatbot reside en el **entrenamiento** que se haga del mismo (intenciones, entidades, diálogos)

Sin embargo el **lenguaje natural** no es natural para ninguna máquina, ya que es demasiado **desestructurado** y **ambiguo**. En consecuencia, el procesamiento del lenguaje natural (NLP) es **computacionalmente costoso**. En contraposición, la curva de aprendizaje para usar una **API conversacional** es mucho **más** **simple**, ya que la estructura y la previsibilidad de las API REST proporcionan eficiencia y estabilidad.

Como consecuencia de lo anterior, la evolución de las plataformas chatbot y las plataformas de procesamiento de lenguaje natural han ido de la mano, pero mientras que las primeras están impulsadas principalmente por los proveedores de plataformas de mensajería como Facebook o WeChat, los principales avances en las tecnologías de procesamiento de lenguaje natural provienen de la **plataforma en la nube** y de **proveedores** de servicios como **Google, IBM o Amazon** que ofrecen sus servicios a través de APIs conversacionales.

Como puede observarse en la siguiente figura, en los dos últimos años se ha producido una gran proliferación de plataformas proveedoras de servicios de procesado de lenguaje natural que se ofrecen a través de una API en la nube, y encontramos que todas las grandes empresas tecnológicas han lanzado sus productos para competir en este mercado de soluciones NLP.

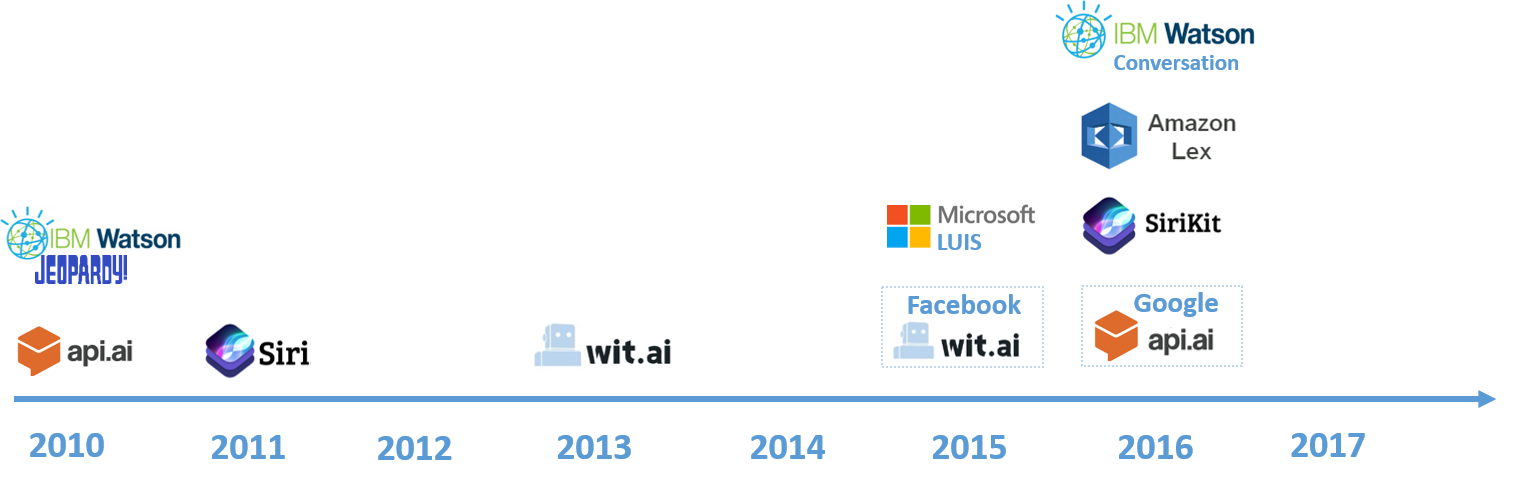


Figura : Evolución temporal principales APIs de NLP

A continuación se presentan las principales características de los proveedores de servicios NLP más relevantes en la actualidad: Alexa (Amazon), Api.ai (Google), Luis (Microsoft), Sirikit (Apple), Watson (IBM) y Wit (Facebook).

Antes de comenzar veamos algunos conceptos básicos de los sistemas de diálogo de lenguaje natural, y que se repetirán a lo largo del estudio comparativo que se realizará en este aparatado. Veámoslo con un ejemplo, supongamos la siguiente frase “*Quiero reservar un vuelo de Madrid a Sevilla para mañana a las 15 horas*”:

* **Intención** (¿Cuál es el objetivo u objetivos de nuestro chatbot?)
  + Propósito de la entrada de un usuario.
  + Las intenciones pueden tener opcionalmente argumentos llamados *slots*
  + En el ejemplo anterior, supongamos que nuestro chatbot gestiona llamadas de una agencia de viajes y que puede realizar funciones como informar sobre horarios, reservar vuelos y cancelar vuelos. En este caso, la intención es “*reservar vuelo*”
* **Locución** (¿Cómo transmite el usuario la intención?)
  + Frases escritas o habladas que invocan la intención. Expresiones de muestra.
  + En el ejemplo anterior la locución sería la frase, tal cual la ha expresado el usuario. Otra posible locución sería “*Quiero un vuelo para Sevilla*” o “*Quiero ir mañana a Sevilla desde Madrid*”, etc.
* **Slot o entidad** (¿Qué información necesitamos para lograr el objetivo?)
  + Información de entrada necesaria para ejecutar la intención. Puede ser obligatorio u opcional.
  + Término u objeto que es relevante para la intención
  + En el ejemplo, los slots podrían ser origen, destino, fecha y número de billetes, todos ellos obligatorios, porque si falta alguno de los datos no se podría ejecutar la acción que indica la intención.

### Conversation (IBM Watson)

* Fue lanzado por **IBM** en **2016,** aunque en **2010** ya se había dado a conocer sus habilidades al proclamarse vencedor frente a 2 dos humanos en el concurso Jeopardy.
* El Servicio de conversación[[1]](#footnote-1) de Watson (WCS) puede definir aspectos de **NLP** como **intenciones**, **entidades** y simular **conversaciones** enteras. Está construido sobre una red neuronal (mil millones de palabras de Wikipedia) que entiende intenciones, interpreta entidades y diálogos.
* Incorpora técnicas de **expansión lingüística** (fuzzy matching) que mejoran sustancialmente su rendimiento, incluso con datos de mala calidad (caso más común). Aunque estas técnicas no funcionan con la misma efectividad en todos los idiomas.
* Trabaja con **slots** (datos de entrada para satisfacer la intención)
* Dispone de una **intefaz gráfica** para la **definición** de **entidades** e **intenciones**, y también se pueden importar y exportar a csv. Existe una interfaz gráfica para la definición del **flujo** de la **conversación**, aunque en este caso es poco manejable, no es intuitiva, no permite deshacer cambios, ni llevar control sobre ellos, salvo exportando el espacio de trabajo a JSON y versionando dicho JSON.
* La **configuración programática** también es posible si no se desea utilizar la interfaz gráfica**.**
* WCS se usa generalmente junto con otros servicios de Watson como Natural Language Understanding o Knowledge Studio. O con Personality Insight y Tone Analyzer para hacer un análisis de personalidad y de sentimientos.
* Para incorporar la **voz** a la conversación se puede utilizar en conjunto con los servicios **Speech To Text y Text To Speech** que realizan transcripción de voz a texto, y síntesis de voz, respectivamente.
* Soporta **9 idiomas** diferentes: inglés, español, portugués, coreano, japonés, italiano, alemán, francés, árabe. No toda la funcionalidad está soportada en todos los idiomas, inglés es el idioma en el que está implementada toda la funcionalidad disponible. Además, soporta 3 idiomas más de forma experimental, estos son chino, checo y holandés.
* Está accesible a través de la **plataforma de cloud** de IBM, Bluemix. Y se proporciona a través de una **interfaz** HTTP **REST**.
* Está disponible de **forma gratuita** por un mes aunque con algunas **restricciones** y limitaciones de uso y respuesta. La **duración** estándar de la cuenta gratuita es de un **mes**, pero para algunas comunidades como por ejemplo la universitaria estas cuentas se extienden considerablemente en el tiempo (entre 6 y 12 meses). También dispone de **cuentas de pago por uso** y de **cuentas** **Premium**.
* Existe bastante **documentación** en inglés, y multitud de ejemplos tanto en la web de IBM con en Github.
* Proporciona **SDK**s para diferentes lenguajes de programación y sistemas operativos, como son Node, Java, Python, iOS, Android, etc. También puede usarse la API de forma nativa o usando curl.
* Puede ser desplegado en **diferentes plataformas o canales** como por ejemplo Facebook Messenger, Slack, Twitter, etc.
* Su **rendimiento** de acuedo con un estudio realizado por Midbowser y Chatbots Journal [3], IBM Watson es la primera elección para la construcción de chatbots para el 61% de las empresas.

### Lex (Amazon)

* Fue lanzado por **Amazon** en **2016**.
* Está construido con las mismas tecnologías de Deep learning que se utilizaron para Alexa.
* Amazon Lex es un servicio[[2]](#footnote-2) para crear interfaces de conversación en cualquier aplicación con voz y texto.
* Proporciona capacidades de **reconocimiento del habla** (convierte voz en texto) y **comprensión del lenguaje natural** para reconocer la intención del texto.
* Trabaja con **slots.**
* Para crear un chatbot, primero se debe definir las intenciones. Por cada intención, se debe añadir ejemplos de locuciones. Se debe definir el flujo de la conversación, y por último, la lógica de negocio necesaria para ejecutar la acción o respuesta.
* Disponible con tarifa de **pago por uso**.
* Accesible mediante **API REST**.
* **Integración** sencilla con muchos otros servicios de la plataforma de AWS:
  + AWS Lambda: para ejecutar código sin aprovisionar ni administrar servidores.
  + Amazon Cognito: para autenticar a los usuarios.
  + Amazon Polly: convierte el texto en voz.
  + AWS Mobile Hub: puede utilizarse para aprovisionar automáticamente bots.
* Soporte **multiplataforma**, se puede usar Amazon API Gateway y AWS Lambda para la integración de chatbots con los siguientes servicios de mensajería como Facebook Messenger, Slack y Twilio SMS. Aunque usando AWS Lambda se puede integrar con cualquier otro sistema de mensajería que tenga una API.
* **SDKs** soportados para creación de bots: iOS, Android, Java, JavaScript, Python, CLI, .NET, Ruby on Rails, PHP, Go y CPP.
* **Sólo** disponible en **único idioma, inglés** estadounidense.

### Api.ai (Google)

* Fue **lanzada** en **2010** y **adquirido** por **Google** en **2016**.
* Recientemente se ha cambiado el nombre a **Dialogflow**.
* Plataforma[[3]](#footnote-3) con capacidades de NLP y NLU. Permite capacidades tales como **detección de intenciones y entidades, análisis de sentimiento, clasificación de contenido y gráficos de relación**.
* Incluye capacidades tales como **reconocimiento de voz** y un conjunto de herramientas de gestión.
* Trabaja con **slots**, detectando si le falta información necesaria, preguntará por los campos obligatorios hasta que el usuario complete la información. Sin embargo, no puede modelar que una intención se pueda detectar solo si un contexto determinado no está presente.
* Proporciona técnicas de **expansión automática**. Esto permite comprender la variación en las entidades sin que tenga que introducir una lista completa. Por ejemplo, si tenemos una intención que diga "Me gusta @color" y en la entidad de color, tenemos azul, verde y rojo. Si un usuario dice "Me gusta el marrón", comprenderá que "marrón" es un color.
* Proporciona **webhook**, y se puede decidir qué intentos van a llamar al webhook, esto puede resultar útil para cambiar los contextos y la respuesta del chatbot, porque se podría pasar la información de la intención detectada a un servicio web y obtener un resultado del mismo.
* Proporciona integración con **diferentes plataformas de mensajería** como Facebook Messenger, Slack, Twitter, Telegram, etc. Es particularmente popular dentro de la comunidad Slack.
* Disponible en Google Cloud a través de una **API REST**.
* Plataforma **gratuita**, solo necesita una cuenta Google.
* Proporciona **SDK** para **diferentes** sistemas operativos y lenguajes de programación: Android, iOS, Cordova, HTML, JavaScript, Node.js, .NET, Unity, Xamarin, C++, Python, Ruby, PHP, Epson Moverio, Botkit y Java.
* Soporta **13 idiomas**: portugués brasileño, chino, inglés, holandés, francés, alemán, italiano, japonés, coreano, portugués, ruso, español y ucraniano.
* Existe gran cantidad de **documentación** y ejemplos de la comunidad de desarrolladores.
* Integración: es posible interactuar con otros sistemas, por ejemplo
  + Es posible conectarla con Google Home creando acciones.
  + También puede integrarse con otros productos de la competencia como Alexa o Cortana.

### Luis (Microsoft)

* Fue **lanzada** por **Microsoft** en **2015**
* El Servicio[[4]](#footnote-4) de Inteligencia de Comprensión del Lenguaje de Microsoft (LUIS) es un componente de los Servicios Cognitivos de Microsoft (MCS) enfocados en crear y procesar modelos de lenguaje natural. Permite crear aplicaciones inteligentes que puedan comprender el lenguaje humano y reaccionar en consecuencia a las solicitudes de los usuarios.
* Trabaja con **intenciones**, **entidades** y el desarrollador debe entrenarlo con ejemplos de enunciados para detectar las intenciones.
* Proporciona una **interfaz de usuario** para ayudar a los desarrolladores a crear intenciones, entidades, pero no proporciona una representación visual de los flujos de conversación. Tampoco permite administrar los parámetros de contexto a través de la interfaz de usuario.
* Proporciona un conjunto de **modelos de idioma ya preparados** que se pueden usar directamente. Incluye un conjunto de entidades preconstruidas. El comportamiento de las entidades preconstruidas no se puede modificar. Algunas de estas entidades son números de teléfono, emails, urls, enciclopedia, geografía, edad, dinero, temperatura, hora, números, ordinales, dimensiones, edad, etc.
* LUIS también admite las **entidades compuestas**. Se trata de una agrupación de entidades en una sola entidad predefinida. Por ejemplo, en "*2 billetes de adultos en primera clase para el vuelo LA hasta NY*". 2 es el número de billetes, *adulto* es el tipo de billete, y *primera* es la clase del billete, pero ambas entidades (tipo de billete y clase) forman parte de la entidad compuesta *compra de billetes*.
* LUIS proporciona **aprendizaje activo**, porque examina todas las palabras con las que ha sido entrenado y para las que tenga una mayor incertidumbre y solicita ayuda para que sean etiquetadas.
* Se puede integrar con el servicio de **reconocimiento de voz** de **Microsoft Cognitive Service** para añadir voz a la conversacion. Se puede usar junto con otras API de procesamiento de texto en MCS, como el análisis lingüístico y el análisis de texto.
* Soporta **10 idiomas** diferentes: inglés, francés, italiano, alemán, español, portugués, japonés, coreano, chino y holandés.
* Dispone de **cuentas gratuitas con restricciones** y 170 euros de crédito. Cuentas para empresas y cuentas de **pago por uso**.
* Está accesible a través de Microsoft Azure mediente una interfaz **API REST**.
* Los **dominios** para crear **bots** son mas **restringidos** que en otras plataformas de la competencia, requiere la creación de aplicaciones UWP (Plataforma Universal de Windows)
* Se puede **configurar programáticamente.**

### Wit.ai (Facebook)

* Fue lanzada en **2013**, y adquirida por **Facebook** en **2015**.
* Wit.ai es la plataforma[[5]](#footnote-5) detrás de las capacidades de **NLP y NLU** del servicio de mensajería Facebook Messenger.
* Proporciona una **interfaz gráfica** de usuario para ayudar a los desarrolladores a crear **intenciones y entidades**.  También consta de una GUI de desarrollador que incluye una representación visual de los **flujos de conversación**, invocaciones de lógica de negocios, variables de contexto, saltos y lógica de bifurcación.
* Soporta **entidades jeráquicas**: por ejemplo, en la frase "*de Los Ángeles a Nueva York*", tanto *LA* como *Nueva York* son entidades de ubicación, pero puede distinguir aún más entre una ubicación de destino (LA) y una ubicación (Nueva York).
* **No soporta slots**, por lo que debe invocar la lógica de negocio después de cada interacción para recopilar cualquier información faltante que el usuario no haya mencionado.
* Proporciona dominios o **entidades reutilizables** predefinidos.
* Proporciona además un conjunto de **herramientas para monitorizar las interacciones** entre los usuarios y la plataforma.
* Soporta **11 idiomas** actualmente (inglés, francés, alemán, italiano, holandés, español, polaco, sueco, portugués, estonio y ruso) aunque en beta soporta 28 idiomas más.
* Está accesible a través de una **interfaz** HTTP **REST**.
* **SDKs** para los siguientes lenguajes de programación: [Node.js,](https://github.com/wit-ai/node-wit) [Python,](https://github.com/wit-ai/pywit) Ruby, etc.
* Wit.ai es **gratis**, incluso para uso comercial, y sin límite de peticiones.
* La **integración** con otros sistemas de mensajería **no están claros**, todos los ejemplos documentados invocan mensajería de **Facebook Messenger**.

### SiriKit (Apple)

* **Siri** es el **sistema de reconocimiento de voz** de **Apple** que utiliza [procesamiento del lenguaje natural](https://es.wikipedia.org/wiki/Procesamiento_del_lenguaje_natural) para responder preguntas, hacer recomendaciones y realizar acciones mediante la delegación de solicitudes hacia un conjunto de [servicios](https://es.wikipedia.org/wiki/Servicio_web) que ha ido aumentando con el tiempo.
* Fue **lanzado** en el **2011**, pero hasta **2016** no se ha **liberado** **Sirikit**, una interfaz para que la comunidad de desarrolladores pueda integrar algunas de sus funcionalidades en sus aplicaciones, con anterioridad solo era posible usarla con las aplicaciones que venían instaladas en el sistema.
* La principal diferencia y limitación con respecto al resto de plataformas que estamos evaluando es que SiriKit **no es** unaplataforma de procesamiento de lenguaje natural **de propósito general**.
* SiriKit está **restringido** **a** un conjunto de **dominios** y tareas particulares con flujos de interacción claramente definidos. Los dominios permitidos son: mensajería, llamadas de voz sobre IP, pagos, ejercicios de entrenamiento, búsqueda de fotos, encargar un recorrido (es decir, pedir un Uber, un taxi…), carplay, etc. Algunas de las aplicaciones que utilizamos con frecuencia no están cubiertas, por ejemplo música, podcasts, estadísticas deportivas, correo electrónico, recordatorios, listas de tareas, etc.
* Dentro de cada dominio hay un conjunto de intenciones que representan las tareas específicas que el usuario puede lograr con Sirikit. Por ejemplo, dentro del dominio de mensajería, existen intenciones de enviar un mensaje, buscar mensajes y establecer atributos en un mensaje. Sirikit **determina la intención** del usuario y decide si una consulta de voz en particular debe manejarse por sí misma o enviarse a una aplicación de un tercero.
* Toda experiencia de SiriKit se compone de cuatro pasos:
  + **Habla**: es traducida a texto para capturar las instrucciones.
  + **Intención**: se genera desde el habla y define qué hemos de realizar.
  + **Acción**: aquello que se envía a través de una extensión a la aplicación que tiene que generar dicha acción.
  + **Respuesta**: aquella que la aplicación devuelve, y que Sirikit nos trasladará como feedback de vuelta a nuestra acción.
* Para hacer todo este proceso posible Apple ha creado dos nuevos tipos de extensiones para el sistema:
  + **Extensión *Intent***: controlador o vía de comunicación entre la aplicación y Sirikit. Es posible tener varias instancias. Implementa los métodos de resolución, respuesta y manejo de todo el proceso
  + **Extensión *Intent UI***: proporciona una experiencia de usuario personalizada en cada una de las respuestas que partan de nuestra aplicación hacia Sirikit.
* Soporta **8 idiomas**: inglés, alemán, francés, japonés, español, ​ italiano, chino y coreano.
* No está claro el coste por uso, parece que se ofrece de forma gratuita para las aplicaciones que estén dentro de los dominios de uso de Sirikit y que acepten la política de uso.

### Criterios de selección

#### Rendimiento de la tecnología NLP

Para la evaluación del rendimiento de las plataformas bajo estudio nos hemos basado en el trabajo realizado en agosto de este año por la compañía Norte Americana *Intento* [2].

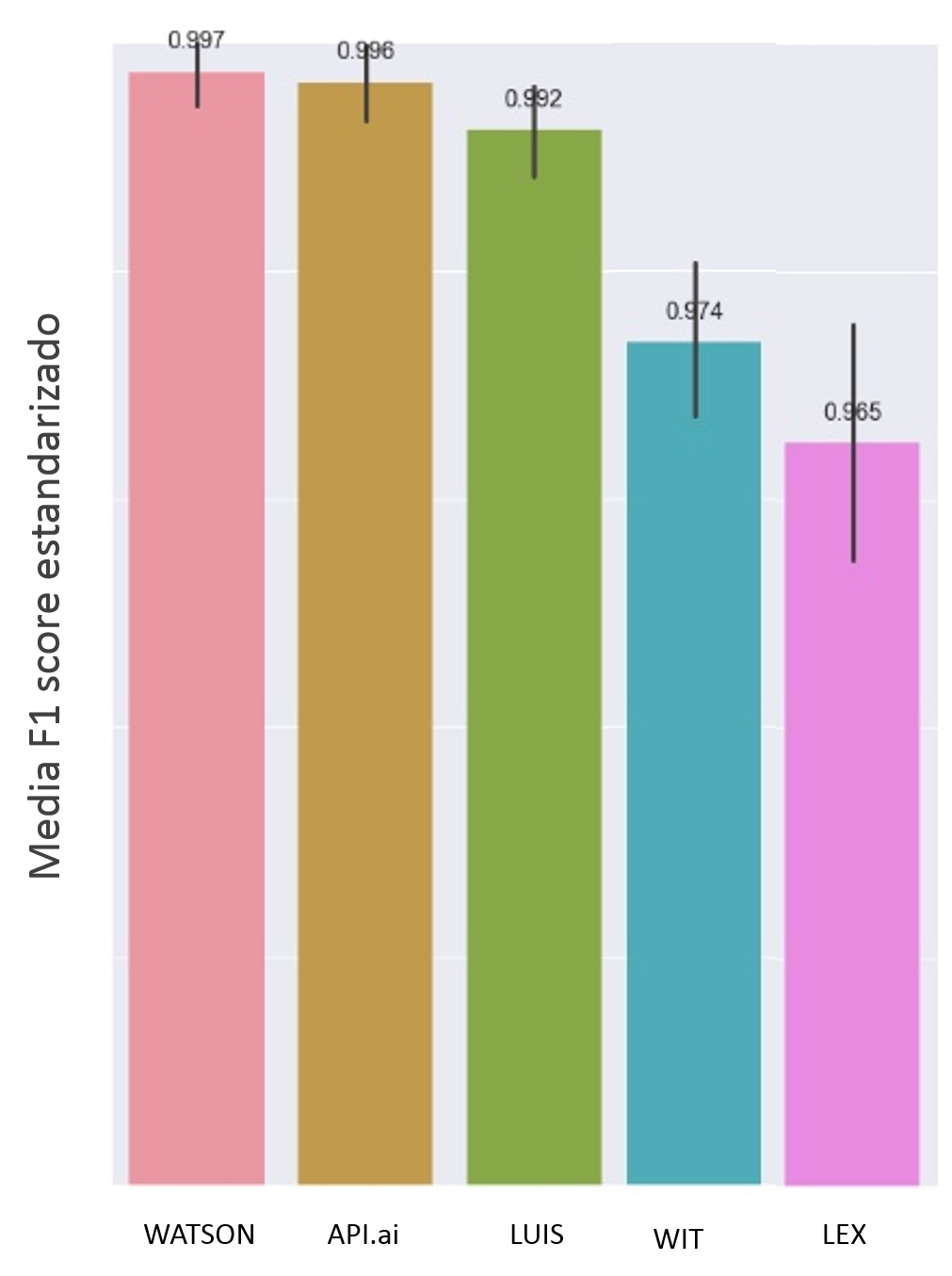
En dicho estudio se han utilizado datos que cubren la mayor variedad de casos. Dichos datos están disponibles públicamente en Github[[6]](#footnote-6).

Algunas cuestiones importantes de este estudio:

* Características del dataset utilizado: 7 intenciones (consultar el tiempo, reservar un restaurante, poner música, añadir canción a playlist y buscar cartelera), 15.600 ejemplos y 340 símbolos.
* Para Amazon Lex ha sido necesario reducir el conjunto de entrenamiento porque la API tiene una limitación de 200.000 símbolos.
* La evaluación se ha realizado en un idioma: inglés.

A continuación se resumen las métricas de rendimiento (predicción de intenciones, falsos positivos, tiempo de aprendizaje y tiempo de respuesta) obtenidas para las plataformas evaluadas.

##### **Predicción de intenciones**



\*Las barras negras indican el intervalo de confianza

Figura : Precisión de detección

La plataforma que presenta mayor precisión es Watson, seguida muy de cerca de API.ai y LUIS. Si enfrentamos precisión y sensibilidad obtenemos la siguiente gráfica:

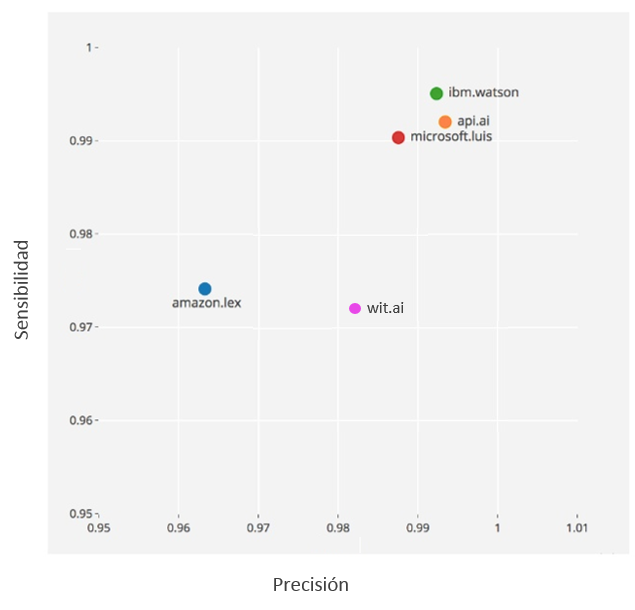


Figura Precisión vs Sensibilidad

Nuevamemente Watson obtiene mejor relación precisión / sensibilidad, seguido muy de cerca de API.ai y LUIS. Desde del punto de vista de la precisión en la detección de intenciones, las mejores plataformas son IBM Watson, API.ai y Microsoft LUIS.

Hay intenciones, como por ejemplo la del pronóstico meteorológico (getWeather), para la que todas las plataformas funcionan bien, pero para las intenciones más complicadas de detectar tener un buen producto es importante.

##### **Falsos positivos**

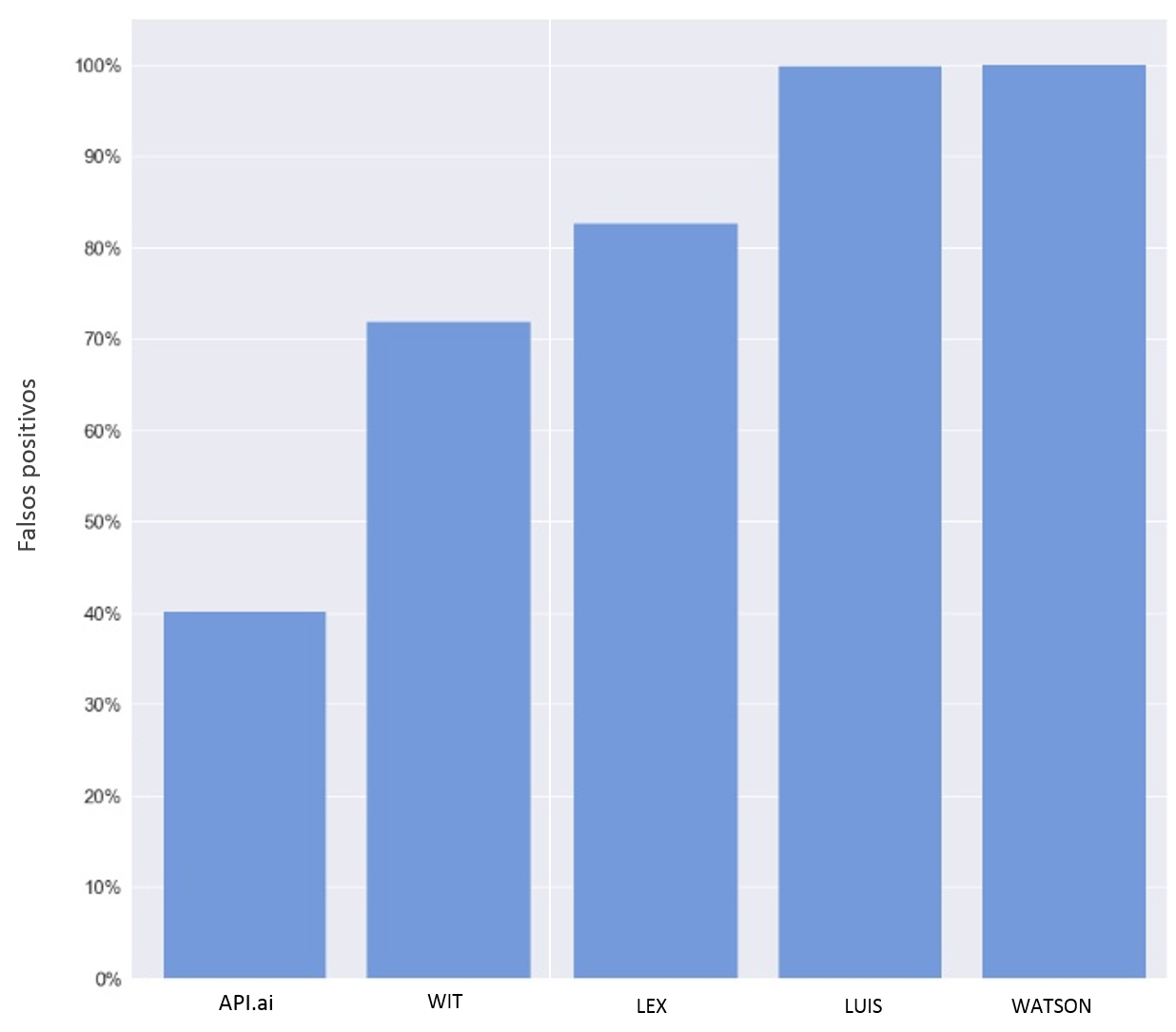


Figura : Porcentaje de falsos positivos

Sólo API.ai es bueno detectando las peticiones del usuario para las que no ha sido entrenado. IBM Watson y Microsoft LUIS intentan mapear la petición de usuairo con alguna de las intenciones del conjunto de datos con el que han sido entrenados.

##### **Velocidad de aprendizaje**

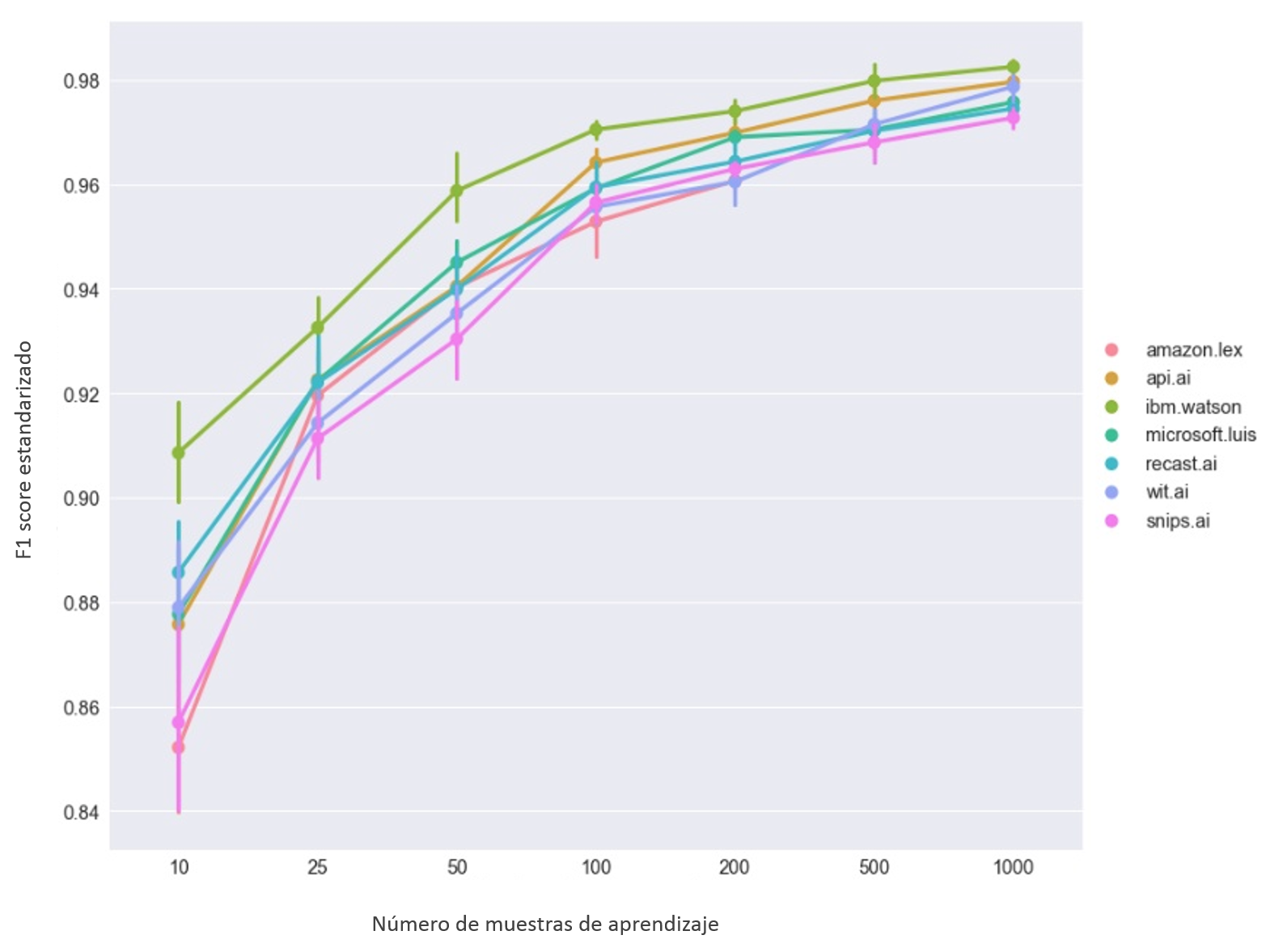


Figura : Velocidad de aprendizaje para cada plataforma

Observamos que la plataforma que presenta mayor velocidad de aprendizaje es Watson. Especialmente para muestras de entrenamiento menores a 100 locuciones, donde IBM Watson es bastante superior al resto de plataformas. Para muestras mayores que 100 locuciones API.ai se aproxima mucho a Watson.

##### **Tiempo de respuesta**

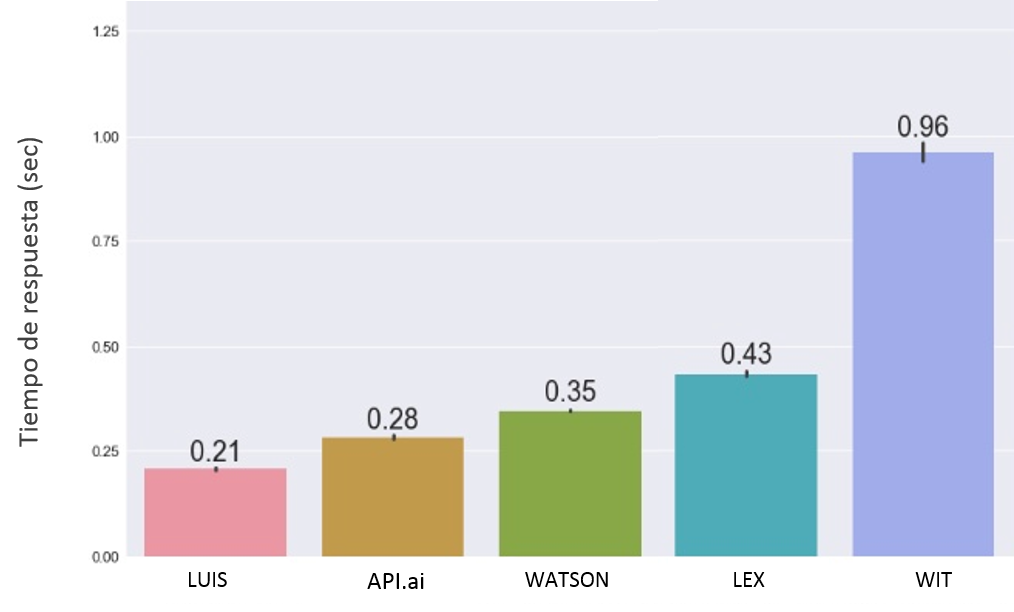


Figura : Tiempo medio de respuesta (segundos)

En este caso, respecto a la velocidad de respuesta Wit.ai es la que obtiene obtiene mucho mejores resultados que sus competidores. Sin embargo, veamos cómo se posiciona cuando comparamos tiempo de respuesta y precisión.



Figura : Rendimiento vs velocidad de respuesta

#### Cobertura de idiomas

* **Watson (IBM)**: soporta **9** idiomas diferentes (inglés, español, portugués, coreano, japonés, italiano, alemán, francés, árabe) y **3** idiomas **experimentales** (chino, checo y holandés).
* **Lex (Amazon)**: sólo disponible en **1 único idioma**, el inglés estadounidense.
* **Api.ai (Google)**: soporta **12** idiomas (portugués, chino, inglés, holandés, francés, alemán, italiano, japonés, coreano, ruso, español y ucraniano). También soporta idiomas locales como el chino cantonés, francés canadiense, inglés US, inglés UK, inglés candiense, inglés australiano o portugués brasileño.
* **Luis (Microsoft)**: soporta 10 idiomas (inglés, francés, italiano, alemán, español, portugués, japonés, coreano, chino y holandés).
* **Wit.ai (Facebook):** soporta **11** idiomas actualmente (inglés, francés, alemán, italiano, holandés, español, polaco, sueco, portugués, estonio y ruso) aunque en **beta** soporta otros **28** idiomas más.
* **Sirikit (Apple):** soporta **8** idiomas (inglés, alemán, francés, japonés, español, italiano, chino y coreano)

Salvo excepciones como **Lex de Amazon** que presenta **serias** **limitaciones** en cuanto a soporte de idiomas (sólo está disponible para inglés), el resto de plataformas soportan todos los idiomas mayoritarios. **Destacan** especialmente por su cobertura **Wit.ai y Api.ai**. Será necesario profundizar en el estudio de la cobertura lingüística real de cada idioma, puesto que para inglés todas parecen tener su óptimo de rendimiento, mientras que para el resto de idiomas, no está tan experimientado y a medida que profundizamos en su uso encontramos limitaciones funcionales o una **disminución del rendimiento** **con** **respecto** al **proporcionado** para **inglés**.

#### Precio del servicio

* **Watson (IBM**): Dispone de cuentas de prueba gratuitas, cuentas de **pago por uso** y cuentas Premium.
  + Las **cuentas gratuitas** tienen restricciones funcionales (límitado a 25 intenciones y 25 entidades) y de uso (límite de 10.000 llamadas a la API al mes). Por defecto tienen una duración de un mes, pero para la comunidad universitaria o para los desarrolladores de hackathones o de sus programas de marketing tienen una duración que oscila entre 6 y 12 meses.
  + **Pago por uso**: el servicio **Convesation** tiene un coste de 0.0025 USD por cada llamada a la API, sin límite de llamadas. Si se añade reconocimiento y síntesis de **voz** a la aplicación, el precio se eleva porque hay que añadir los costes del uso de los servicios **Speech To Text** (1.000 minutos gratis al mes y a partir de ahí los precios oscilan entre 0.02 USD y 0.01 USD por minuto según tramos de consumo) y **Text to Speech** (1 millón de caracteres gratis al mes, a partir de ahí la tarifa es de 0.02 USD por cada 1.000 caracteres que se solicite sintetizar).
* **Lex (Amazon)**: dispone de cuentas de prueba y cuentas de **pago por uso** (cantidad de solicitudes de voz y texto procesadas). El precio por solicitud de **voz** es de 0,004 USD y el precio de cada solicitud de **texto** es de 0,00075 USD.
* **Api.ai (Google):** gratuito.
* **Luis (Microsoft)**: dispone de cuentas gratuitas con restricciones y 170 euros de crédito. Cuentas para empresas, y cuentas de **pago por uso** con un coste de 0.000633 USD por llamada a la API, siendo las 10.000 primeras solicitudes del mes gratuitas.
* **Wit.ai (Facebook)**: gratuito.
* **Sirikit (Apple):** No está claro el coste por uso, parece que se ofrece de forma gratuita para las aplicaciones que estén dentro de los dominios de uso.

Es difícil estimar una cifra final con la que comparar para las plataformas de pago por uso, ya que cada una de ellas tiene una tarificación distinta y en la arquitectura de la solución final puede que sea necesario añadir costes adicionales como el de añadir o no voz.

El número de llamadas a estas APIs dependerá en gran medida del número de usuarios, y del tipo de peticiones que estos realicen.

Además no solo hay que contar el consumo una vez la aplicación está en producción, sino que en el proceso de construcción, entrenamiento y pruebas antes de lanzarla también será necesario realizar peticiones a la API y esto conlleva también un coste.

Con respecto a este criterio de selección parece que la plataforma que destaca es Api.ai en relación calidad precio, puesto que ofrece uno de los mejores rendimiento y además es gratuita.

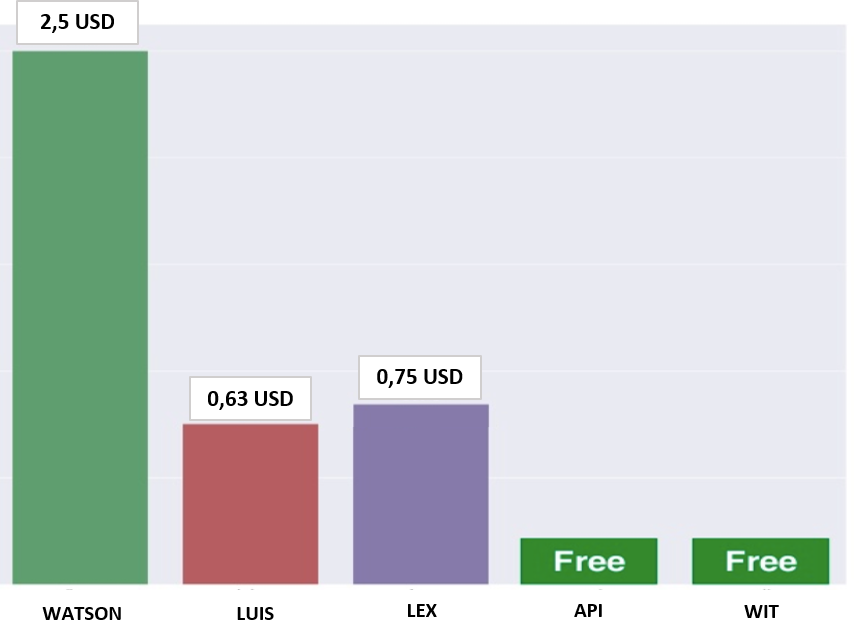


Figura : Precio (USD) por cada 1K peticiones de texto

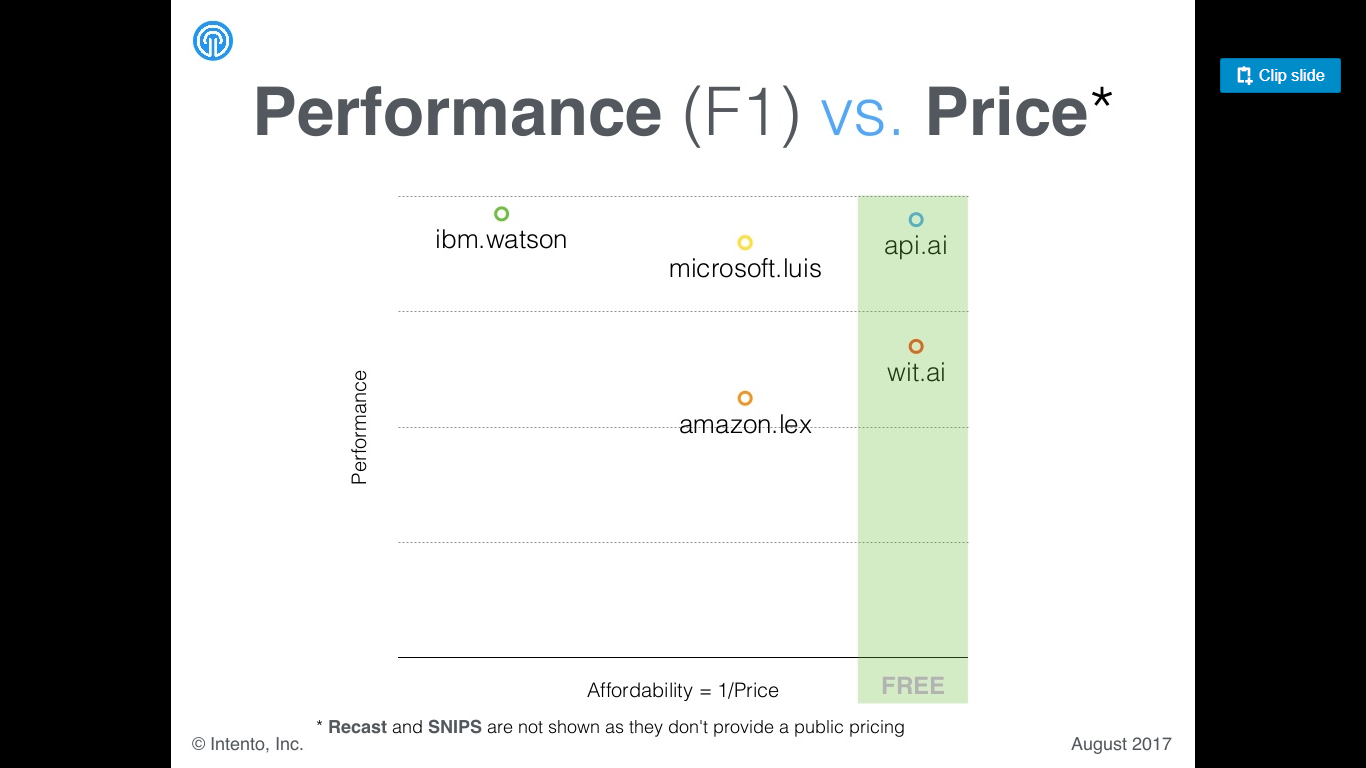


Figura : Rendimiento vs precio

#### Fortaleza de la compañía proveedora

Todas las compañías proveedoras que estamos evaluando son gigantes tecnológicos, fuertes y consolidados. Rivales que compiten por el dominio tecnológico, el mercado y los clientes. Cada uno de ellos compite con su estrategia y con campañas de marketing en las que han usado la inteligencia artificial para hacer sus sistemas publicitarios más efectivos.

**IBM ganó notoriedad** por haber salido victorioso con Watson frente a 2 humanos en el concurso **Jeopardy**, pero en los años siguientes, el espacio de inteligencia artificial, y más concretamente el procesado de lenguaje natural, ha visto entradas importantes de compañías como **Google y Amazon**, que tienen una **gran cantidad de datos** de usuarios y clientes para alimentar sus sistemas.

El modelo de negocio que se está construyendo alrededor del procesado de lenguaje natural, reconocimiento vocal pasa por ofrecer este como un **servicio en la nube**, y actualmente la nube está **dominada** **por Amazon Web Services** y **Microsoft Azure**.

#### Existencia de módulos predefinidos

Evaluaremos en esta sección si la plataforma dispone de módulos genéricos o de uso común ya precargados en el sistema para que el desarollador pueda arrancar con más rapidez.

* **Watson (IBM)**: no dispone de intenciones pre-construidas, lo más parecido son las variables de sistema definidas para manejar por ejemplo monedas, fechas, horas, localizaciones, número, porcentajes, etc., y en beta aún para persona[[7]](#footnote-7). También permite importar intenciones o entidades en formato csv, esto se puede aprovechar para incorporar intenciones genéricas como las de saludar, despedirse, etc.
* **Lex (Amazon)**: La librería *Alexa Skill Kit* proporciona intenciones preconstruidas para acciones generales, como detener, cancelar y pedir ayuda. También existen entidades ya construidas para los datos más comunes como emails, porcentajes, números de teléfono, unidades de medida, fechas, horas, etc[[8]](#footnote-8).
* **Api.ai (Google):** proporciona entidades predefinidas para fechas, horas, números, colores, etc. con el objetivo[[9]](#footnote-9)
* **Luis (Microsoft)**: Proporciona una amplia gama[[10]](#footnote-10) de entidades preconstruidas disponibles en la mayoría de idiomas. Tambien incorpora dominios predefinidos que proporcionan entidades e intenciones, algunos de estos dominios son calendario, cámara, comunicación, música, etc. Y por último, también permite utilizar las intenciones y entidades de Cortana en nuestra aplicación.
* **Wit.ai (Facebook)**: tiene sus modelos preexistentes y preconstruidos que pueden reconocer entidades e intenciones, como lugares, correos electrónicos, calendarios, etc.
* **Sirikit (Apple):** no se han encontrado elementos predefinidos, peroal tratarse de una plataforma con dominios más restringidos, el usuario no tendrá que proporcionar tantos ejemplos a la hora de entrenar para que la aplicación comience a funcionar con las intenciones específicas.

#### Control de versiones

El control de versiones es importante y más aún cuando se trabaja en equipo. Resulta imprescindible tener archivados los cambios que se realizan sobre nuestro chatbot, asi como las intenciones, entidades y el flujo de la conversación que hemos definido para él, con un histórico de los cambios que se han realizado a lo largo del tiempo para poder recuperar versiones específicas más adelante.

Tanto **AWS Lex**[[11]](#footnote-11) como **LUIS**[[12]](#footnote-12) incorporan control de versiones, en el resto de plataformas no hay disponible mecanismos de este tipo, la única opción pasa por llevar el control de forma manual exportando los espacios de trabajo y salvando las distintas versiones que se van generando. Tanto en Wit.ai como en Watson se puede exportar a formato JSON. Este JSON contiene las intenciones, las entidades y el flujo de la conversación que se ha definido para nuestra aplicación.

#### Flexibilidad, autonomía y accesibilidad

Salvo **Sirikit** que está **restringida** a los **dominios** que se enumeraron en el apartado 2.2.6, el resto de plataformas NLP evaluadas trabajan de forma similar, proporcionando flexibilidad para la construcción de chatbots en cualquier campo o ámbito de uso. Permiten usar intenciones y entidades predefinidas asi como crear otras específicas que se adapten mejor al caso de uso para el que el chatbot sea destinado.

#### Integración

Presentan **limitaciones** en este aspecto **Sirikit** que sólo se integra con sistemas iOS. **LUIS** que necesita la creación de aplicaciones UWP (Plataforma Universal de Windows) y **Wit.ai** que no parece integrarse con facilidad en otros sistemas de mensajería diferentes a Facebook.

#### Reconocimiento y síntesis de voz

Muchos chats tienen una función de mensaje de voz. La razón es obvia, escribir con el teclado a veces puede resultar tedioso y es más fácil y rápido expresarse usando la voz. Sin embargo hay ocasiones en que las circunstancias te obligan a guardar silencio y otros en los que necesitas realizar una búsqueda en tu historial de chat para recuperar información que no puedes encontrar en los mensajes de voz. Por estas razones nuestro chatbot debe poder comunicarse tanto en texto como en voz.

Veamos en más detalle cómo se adapta a este requisito las plataformas de NLP que estamos estudiando.

* **Watson (IBM)**: combinado con los servicio **Watson Speech to Text** y **Tex to Speech**, es posible incorporar voz en las conversaciones de nuestro chatbot. Speech to Text transcribe la voz a text y Text to Speech sintetiza la voz dado un texto como entrada. El servicio conversation recibe texto como entrada y devuelve texto.
* **Lex (Amazon)**: Amazon Lex comprende tanto texto como voz, pero si la respuesta de nuestro chatbot es también usando voz, será necesario utilizar un el servicio **Polly** de Amazon, que transforma texto en voz.
* **Api.ai (Google)**: El servicio de reconocimeinto de voz de API.ai está en desuso, para incorporar voz a la conversación hay que incorporar servcios para reconocimiento de voz, y para síntesis de voz. Se puede usar la API **Speech de Google Cloud**, que convierte audio en texto aplicando potentes modelos de redes neuronales en una API fácil de usar. La API reconoce más de 80 idiomas y variantes. No se ha encontrado ninguna API para síntesis de voz en los servicios de la nube de Google.
* **Luis (Microsoft)**: Se puede integrar con el servicio **Bing Speech API** de los servicios cognitivos de Microsoftpara añadir voz a la conversacion. Este servicio convierte voz en texto y viceversa.
* **Wit.ai (Facebook):** Wit.ai entiende tanto texto como voz. Admite voz en tiempo real y audios. No se ha encontrado en la documentación si dispone de métodos en su API para sintetizar voz.
* **Sirikit (Apple):** comprende tanto texto como voz, proporciona un método llamado **AVSpeechSynthesizer** para sintetizar voz.

### Conclusiones

Los chatbots son una tendencia ascendente. Cuando son de calidad pueden aumentar el compromiso con los usuarios, brindar una mejor experiencia y ahorrar costes. Sin embargo, hacerlo bien no es trivial.

Actualmente, hay una gran cantidad de plataformas que pueden ayudar en la creación de chatbots. Algunas de estas plataformas, como las que se mencionan en el apartado 2.1 han surgido para beneficiarse del boom momentáneo del mercado, algunos son muy básicos y cubren casos de uso demasiado específicos, por lo que difícilmente pueden ser escalables. Son buenas soluciones para experimentar con la experiencia del usuario final, pero no para entornos productivos.

Para la construcción de chatbots ambiciosos, capaces de manejar conversaciones complejas y realizar acciones, son más apropiadas, por su estabilidad, rendimiento y flexibilidad plataformas como las mencionadas en el apartado 2.2, y que proveen servicios para incorporar las técnicas de procesado de lenguaje natural en nuestro chatbot.

Watson tiene el mejor rendimiento de detección de intenciones, especialmente en conjuntos de datos de entrenamiento más pequeños. Como desventaja su precio es el más elevado de todas las plataformas evaluadas en este estudio.

Wit.ai es gratuito y tiene la mejor cobertura de idioma, pero presenta un rendimiento y tiempo de respuesta peor que los de sus competidores.

API.ai también es gratuito y tiene uno de los mejores resultados.

LUIS también tiene un rendimiento de alto nivel, y un tiempo de respuesta menor que otros con los que ha sido comparado.

Lex tiene buen rendimiento, pero solo está disponibles en inglés.

Sirikit está restringido sólo a ciertos dominios de uso y no aporta flexibilidad en la construcción de chatbots.

## IBM Watson

### Introducción

IBM Bluemix es la plataforma cloud de IBM que proporciona acceso a desarrolladores móviles y web al software y servicios de IBM.

El entorno de IBM Bluemix es un ecosistema abierto de servicios. Cuenta con un catálogo de servicios como bases de datos, soporte móvil, análisis y seguridad para sus aplicaciones. Servicios de integración

permitir que las aplicaciones accedan a cargas de trabajo tradicionales que se ejecutan en la organización

entorno local.

Bluemix soporta [Java](https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)), [Node.js](https://es.wikipedia.org/wiki/Node.js), [Go](https://es.wikipedia.org/wiki/Go_(lenguaje_de_programaci%C3%B3n)), [PHP](https://es.wikipedia.org/wiki/PHP), [Python](https://es.wikipedia.org/wiki/Python), Ruby, [GeneXus](https://es.wikipedia.org/w/index.php?title=GeneXus&action=edit&redlink=1" \o "GeneXus (aún no redactado)) y puede ser extendido a otros lenguajes tales como [Scala](https://es.wikipedia.org/wiki/Scala_(lenguaje_de_programaci%C3%B3n)" \o "Scala (lenguaje de programación)) mediante el uso de 'buildpacks'.

La gran diferencia es adaptar a nuestro propio contexto, puedo entrenar, pero no puedo alterar los parámetros de los modelos, para eso están otro tipos de servicios de machine learning. La idea de watson es enmascarar los modelos y la funcionalidad que hay por debajo.

Lo que ofrece watson son APIs, siempre voy a necesitar una aplicación o entorno que orqueste estos servicios. Se puede usar cualquier lenguaje como orquestador, porque al final es una llamada a un http.

### Conversation

La siguiente figura muestra la arquitectura general de una solución que integra el servicio de Conversación de IBM Watson



Figura Arquitectura típica de una aplicación de conversación

1. Los usuarios interactúan con la aplicación a través de las interfaces elegidas (servicios de mensajería, ventana de chat dentro de un sitio web, o incluso interfaces de audio cuando se combinan con los servicios Watson Speech to Text).
2. La aplicación envía la entrada del usuario al servicio de conversación.
3. El servicio de conversación interpreta dicha entrada e identifica los conceptos para los que fue entrenado.
4. En función de los conceptos que identifica, dirige el flujo de conversación.
5. Reúne la información que necesita consultando diferentes fuentes de datos.
6. Finalmente proporciona la información al usuario.

La aplicación se conecta a un espacio de trabajo del servicio de conversación. El espacio de trabajo es un contenedor para los artefactos que definen el flujo de conversación y que proporciona la comprensión del lenguaje natural.

Es posible definir múltiples espacios de trabajo en cada instancia de servicio de conversación de Watson.

Cada espacio de trabajo es entrenado en un determinado idioma para reconocer ciertos conceptos y para dirigir el flujo de conversación que rige la interacción con usuario.

#### Intenciones, entidades y diálogo

Watson de IBM trabaja con 3 conceptos: las intenciones, las entidades y los diálogos.

El primero de ellos hace referencia a las acciones que puede querer hacer el cliente (“quiero sacar dinero”).

El segundo, a los datos que necesitaremos para definir la acción y que el bot sepa exactamente que hacer (¿sacar dinero dónde? ¿A mi alrededor, en el cajero más cercano, en el más barato? ¿En otro sitio al que voy a ir dentro de media hora?).

El tercero, el lugar donde definiremos el diálogo, vía nodos y preguntas que se desencadenen cuando el bot necesite más información para saber qué acción llevar a cabo.

#### Fuzzy matching

Puede activar la concordancia difusa para mejorar la capacidad del servicio de reconocer términos de entrada de usuario con una sintaxis similar a la entidad, pero sin requerir una coincidencia exacta.

Hay tres componentes para la coincidencia aproximada: origen, error ortográfico y coincidencia parcial:

* **Stemming**: la función reconoce la forma madre de los valores de entidad que tienen varias formas gramaticales. Por ejemplo, la raíz de 'plátanos' sería 'banana', mientras que la raíz de 'running' sería 'run'.
* **Errores ortográficos**: la función puede asignar la entrada del usuario a la entidad correspondiente correspondiente a pesar de la presencia de errores ortográficos o ligeras diferencias sintácticas. Por ejemplo, si define "jirafa" como sinónimo de una entidad animal, y la entrada del usuario contiene los términos "jirafas" o "jirafa", el emparejamiento difuso es capaz de asignar correctamente el término a la entidad animal.
* **Concordancia parcial**: con la coincidencia parcial, la característica sugiere automáticamente sinónimos basados ​​en subcadenas presentes en las entidades definidas por el usuario, y asigna un puntaje de confianza inferior en comparación con la coincidencia de entidad exacta. Nota: en inglés, la coincidencia difusa evita la captura de algunas palabras comunes válidas en inglés como coincidencias difusas para una entidad determinada. Actualmente, esta característica utiliza solo palabras estándar del diccionario de inglés y no utiliza sinónimos definidos por el usuario.

### Visual recognition

### Speech to Text

### Text to Speech

# Trabajo realizado

A

continuación el trabajo realizado y la metodología empleada. Una vez realizado el estudio teórico de los servicios Watson y las APIs con las que se integrará nuestro asistente, el siguiente paso fue la familiarización práctica con las APIs. La información sobre algunas de ellas, o sobre funcionalidades específicas es escasa por lo que para averiguar su funcionalidad, o el significado de algunos parámetros, fue necesario realizar pruebas con los ejemplos del servicio.

En este proceso se encontraron limitaciones importantes en el servicio *Food2Fork* y en la integración con *Slack*, lo que obligó a realizar una adaptación previa de dicho servicio. Los detalles de estas adaptaciones se explicarán a continuación en los apartados 3.2 y 3.7.

Después de solucionar los problemas mencionados, se procedió a codificar el resto de módulos.

Tras esta introducción, se explicará en más detalle el trabajo realizado. En la primera parte, apartado 3.1 se describe la arquitectura de la solución propuesta.

## Arquitectura de la solución

A continuación la arquitectura a alto nivel del prototipo implementado para el asistente conversacional:

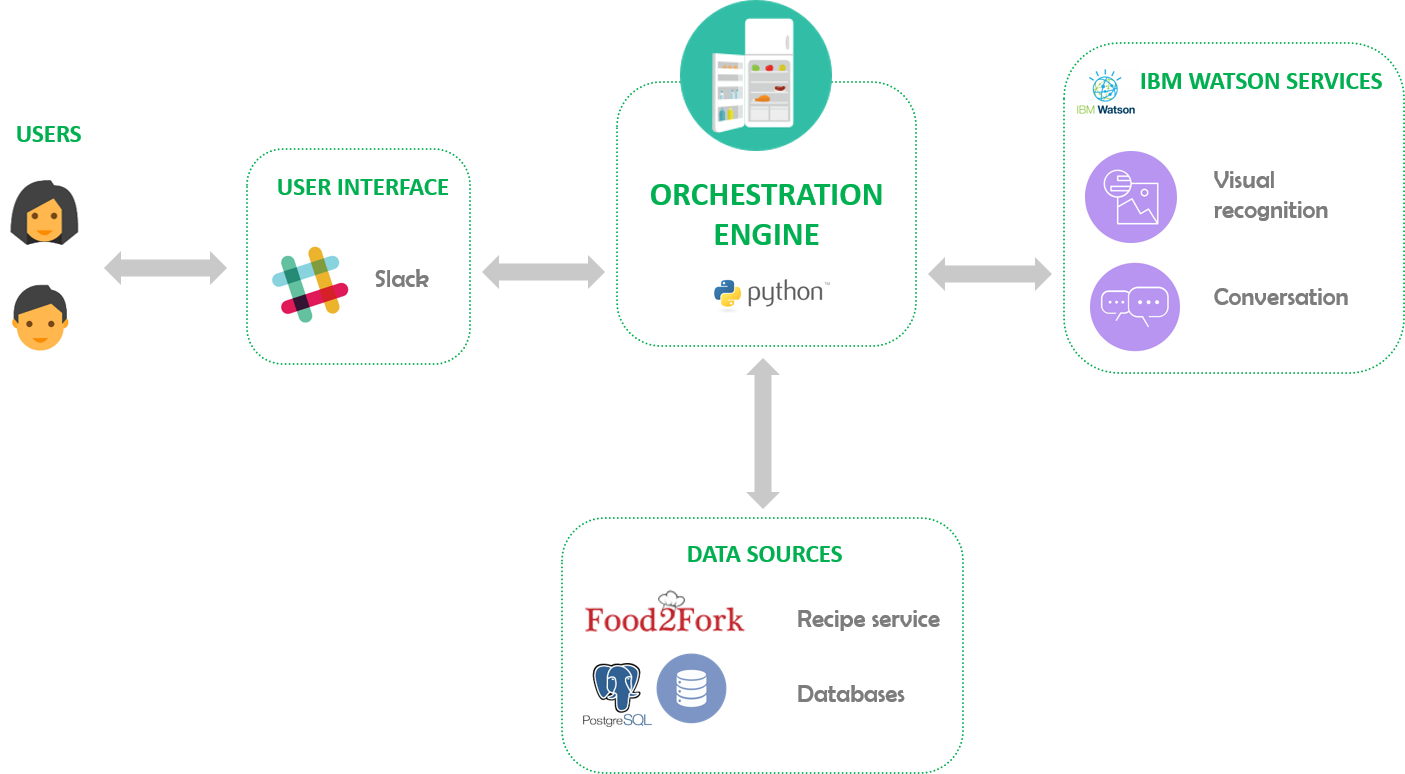


Figura Arquitectura de la solución

Se compone de los siguientes bloques funcionales:

* **User interface:** medio a través del cual los usuarios podrán comunicarse en **tiempo real** con el asistente. La comunicación está soportada por la plataforma de mensajería *Slack* a través WebSockets[[13]](#footnote-13). La comunicación se realizará utilizando la interfaz web de Slack a través de texto, también será posible compartir ficheros con imágenes de comida que el asistente intentará reconocer y proporcionar su receta.

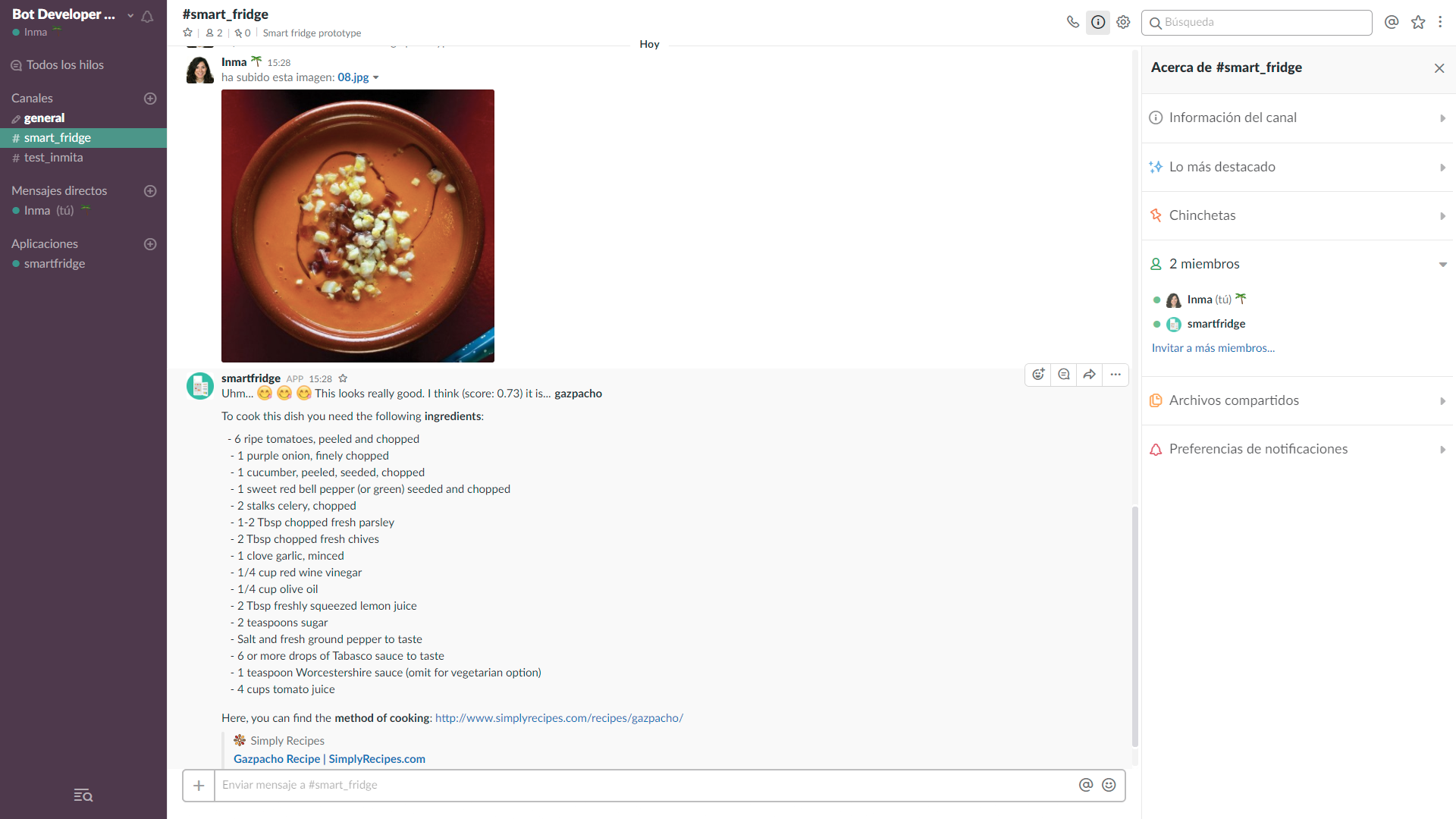


Figura Interfaz usuario: Slack

* **Data sources:** este bloque representa las fuentes de datos que utiliza el asistente para componer la respuesta que ofrecerá al usuario. El prototipo consulta principalmente las siguientes fuentes de datos:
  + **Base de datos local:** base de datos postgres con una tabla en la que se persisten todos los productos que se almacenan en el frigorífico o despensa de tu casa.
  + **Servicio de recetas en la nube** (Food2Fork): proporciona todo tipo de recetas clasificadas socialmente que han sido extraidas previamente de blogs de cocina y otros sitios web.
* **IBM Watson services:** servicios IBM Watson utilizados porel prototipo. Se utilizan los siguientes servicios:
  + **Visual recognition:** identifica y clasifica escenas, objetos, comida, personas, etc. a partir de la imagen que se le proporcione como entrada.
  + **Conversation:** proporciona la comprensión del lenguaje natural y simula una conversación humana.
* **Orchestration engine:** pieza central de la arquitectura, es la encargada de orquestar las peticiones a los servicios, el acceso de base de datos, la comunicación con el usuario en tiempo real y con toda esta información componer y hacer llegar al usurio la respuesta más adecuada a la consulta o petición que realizó.

## Interfaz de usuario

### Selección de plataforma

#### Facebook Messenger

No solo es el gigante de las redes sociales, con 12.000 millones de usuarios activos mensuales, además la firma ha dejado claro en la conferencia F8 de desarrolladores que su apuesta para el futuro está orientada a chatbots

El gigante de las redes sociales, con 12 billones de usuarios activos mensuales a nivel mundial, ya dejó claro en el F8 que su apuesta para el futuro está orientada a chatbots e Inteligencia Artificial. Tanto es así, que acaba de anunciar que será posible lanzar publicidad a través de Messenger. De este modo, las empresas podrán crear campañas y contactar directamente con todos aquellos usuarios que hayan iniciado una conversación. Además, su diseño de interfaz permite botones, texto y un sinfín de opciones que hacen que la experiencia de usuario se adapte al máximo al contexto y las necesidades de la marca.

#### Whatsapp

Es la app de mensajería por excelencia en España y Latinoamérica y, según datos del Eurobarómetro, España es el país de Europa que más la utiliza. A pesar de no ser un bot muy sofisticado, sí que permite ampliar las funcionalidades de Whatsapp. La aplicación se llama qeuBot. Es el equivalente a los bots de Telegram y, básicamente, aglutina diferentes funciones desde el propio chat de Whatsapp.

Es la app de mensajería por excelencia en España y Latinoamérica. Sin embargo, la app con más de 1.3 billones de usuarios mensuales activos es la que más se está demorando en adaptarse al auge de los bots. Por lo que la responsabilidad y costes corren a cuenta de las empresas.

#### Telegram

Se podría decir que es la mejor adaptada a esta nueva tecnología. En este canal encontramos cientos de *bots* que informan sobre ofertas, libros, imágenes, GIF’s, etc. Además, cuenta con un sistema de pago que permite monetizar al *chatbot*, por lo que se convierte en un canal muy atractivo para todas aquellas empresas que quieren vender sus productos a través de esta nueva plataforma. Básicamente han llevado el modelo de compra de iTunes o Google Play a las conversaciones con bots, de forma que con un simple clic el usuario pueda comprar, son los llamados *“In-Chat Payments”*.

Se podría decir que es la mejor adaptada a esta nueva tecnología. En este canal hay cientos de bots que informan sobre ofertas, libros, imágenes, GIF’s… La compañía ha incorporado nuevas funciones para un uso más cómodo y eficaz, entre ellas, Yandex Translate, que ayuda a traducir mensajes de varios idiomas o RT noticias Bot, que envía noticias internacionales.

#### Slack

La multinacional Microsoft y la plataforma Slack son dos de laa compañías que más están apostando en la creación de estos 'softwares'. Slack es una app de mensajería corporativa interna, conecta a empleados y miembros del equipo en un canal de comunicación rápido y fluido. Es una aplicación en la nube que ofrece una combinación de distintos servicios: correo electrónico, mensajería instantánea, etc.

Es la app de mensajería corporativa interna más famosa, conectando a empleados y miembros del equipo en un canal de comunicación rápido y fluido, que permite transferencia de archivos o llamadas de grupo, entre otras muchas funciones. Sus *bots* sirven para dinamizar aún más las gestiones internas en las compañías, por ejemplo, crear incidencias, pedir material, informar sobre vacaciones o reservar una sala de reuniones.

#### Skype

También dispone de un framework o entorno de trabajo que facilita el desarrollo de chatbots. Destaca Ai World Bot, que da acceso al nombre de usuario de Skype y a cualquier mensaje de chat y contenido que los participantes compartan con él. Asistant@Zoom ai. es un asistente virtual personal que permite programar citas, preparar al usuario para ellas, o sugerir presentaciones amenas

Este software también disponen de un framework o entorno de trabajo que facilita el desarrollo de chatbots, aunque cada una de ellas cuentan con un público determinado que la empresa debe conocer antes de decantarse por una u otra. Por ejemplo, Twitter hace poco presentó sus tarjetas para comercializar productos e iniciar conversaciones con los usuarios de forma directa.

### Integración con Slack

Slack es una plataforma de comunicación en equipo que permite la comunicación síncrona y asíncrona en diferentes dispositivos. Integra diferentes servicios en una sola aplicación de mensajería, como por ejemplo Dropbox, Twitter, Github, Google drive, WordPress, etc. De este modo los usuarios de la plataforma pueden disfrutar de todos estos servicios desde un único punto de acceso.

Slack ofrece salas de chat organizadas por temas, así como grupos privados y mensajes directos. También proporciona búsquedas por los chats. La comunicación se pruede realizar a través de texto y también proporciona llamadas de voz y vídeo.

Una de las claves de su triunfo es [su API](https://api.slack.com/) [1], que es actualizada a constantemente para integrar nuevas funcionalidades.

El asistente conversacional que se ha construido utiliza *Slack* como **canal de comunicación**.

Para la integración con Slack la aplicación orquestadora (ver apartado 3.6) utiliza el SDK de Python, concretamente la librería *SlackClient[[14]](#footnote-14).* Se trata de un paquete que facilita el uso de la API de Slack, y que también proporciona conexión con la API RTM (Real Time Messaging) de Slack. Permite por ejemplo administrar las conexiones de mensajería en tiempo real a través de websocket, o gestionar el estado en los canales a los que el usuario bot (token) esté asociado.

Para la integración se han seguido los pasos descritos en los siguientes turoriales [2] y [3]. Básicamente se puede resumir como sigue:

1. Registrarse en Slack web
2. Crea un nuevo usuario bot en slack
3. Obtén el token de acceso a la API
4. Obten el bot ID
5. Descarga el script python de

<https://github.com/mattmakai/slack-starterbot/blob/master/starterbot.py>

1. Configura con los IDs y token de los pasos anteriores
2. Crea un canal público en slack y añade a la aplicación que acabas de crear
3. Ya puedes comenzar a chatear con el bot

## Integración con base de datos

### Conexión

### Modelo de datos

## Configuración y entrenamiento del servicio Conversation

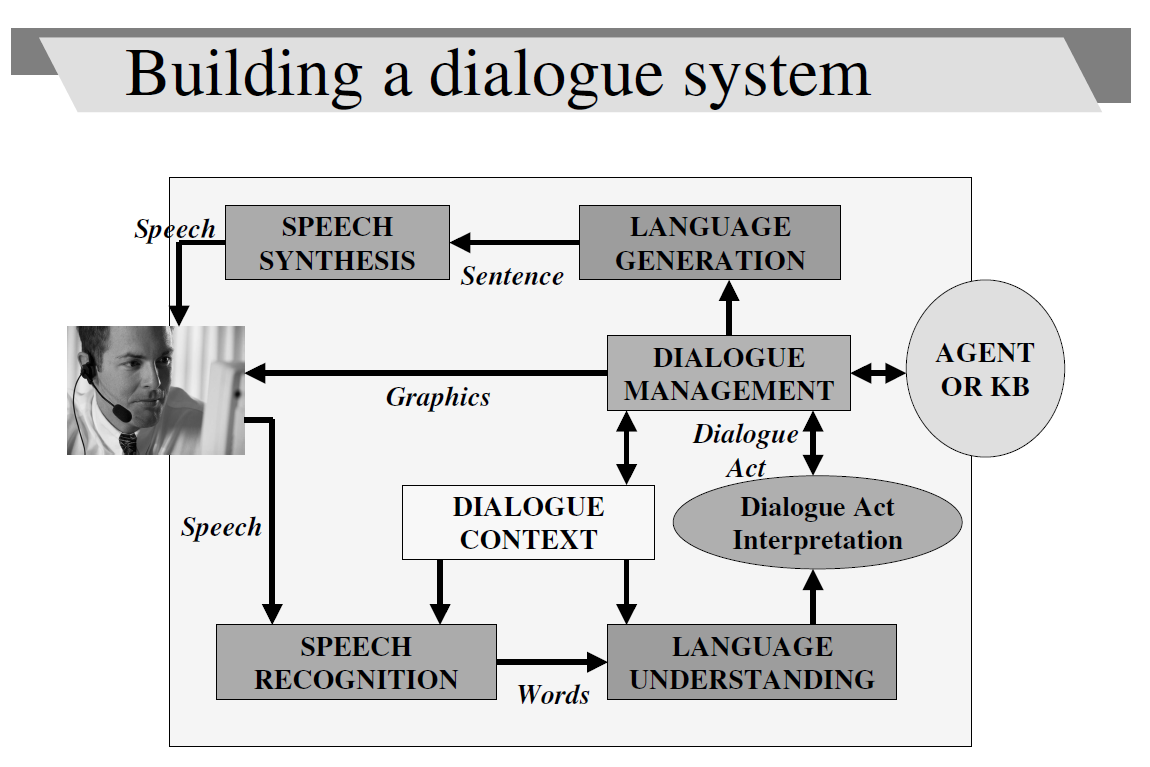
### Intenciones

### Entidades

### Variables de sistema

### Variables de contexto

### Flujo de Diálogo



## Configuración y entrenamiento del servicio Visual recognition

## Implementación de la aplicación orquestadora

## API de recetas de cocina (Food2Fork)

### Selección de la API

A continuación un breve resumen de las APIs evaluadas, no ha sido sencillo encontrar una API gratuita que se adapte a las necesidades. Las que ofrecen más funcionalidad o una mayor riqueza de recetas disponibles no son gratuitas.

Otra limitación importante ha sido que ninguna soporte multiidioma, todas están disponibles en inglés, pero no se ha encontrado ninguna para español.

#### Spoonacular

pide tarjeta, si superas las

50 peticiones al dias

500 resultados

#### Food2Fork

http://food2fork.com/about/api

key=0503617cd3c134aa02ae9a96824bf519

http://food2fork.com/api/search?key=0503617cd3c134aa02ae9a96824bf519&q=shredded%20chicken,tomatoes,cheese,onion,egg

#### Edamam

https://www.edamam.com/recipes/pollo%2C+cebolla%2C+queso

No consigo que responda la API con python

#### Recipe Bridge

solicitado permiso

#### Kraft Recipe

no disponible

#### Pearson Kitchen manager

http://developer.pearson.com/apis/pearson-kitchen-manager/

no funciona

#### Yumly

Pide tarjeta, hay cuenta trial de solo 14 dias

se puede solicitar cuenta de estudiante

https://developer.yummly.com/

#### Bigoven

No proporciona recetas a partir de ingredientes

http://api2.bigoven.com/web/documentation/recipes

#### Recipal

no hace lo que quiero

### Integración

## Adaptación para nuevo caso de uso: asistente para tienda de ropa

# Conclusiones

D

## Resumen del trabajo realizado y cumplimiento de objetivos

## Dificultades, aportaciones y contribuciones

## Resultados: importancia, repercusión y utilidad

Exactitud. Las API generales pueden ser una forma rápida de obtener resultados, pero la precisión de sus algoritmos no puede alcanzar la precisión de los modelos personalizados.

Sorprendió lo poderoso que podría llegar a ser con solo un poco de optimización y entrenamiento para mis casos de uso esperados.

A diferencia de otras áreas de Inteligencia Artificial, las máquinas aún no han alcanzado el nivel de rendimiento humano en lo que respecta a NLU

Tener el caso de uso correcto es clave

Si bien la ciencia detrás de la IA está haciendo un progreso insano, los resultados prácticos no son tan buenos como en las películas de Hollywood.

* Recomendaciones que tengan impacto en el negocio
* Recomendaciones acerca de cómo la adopción del modelo cambaría sus sistemas (ampliando o sustituyendo ciertos módulos y/o actividades)
* Sección centrada en el impacto esperado de la adopción del modelo (según tu opinión).
* Discusión de los potenciales beneficios y riesgos

# Líneas futuras

E

## Comunicación de voz en tiempo real (streaming)

## Integración con Voice Gateway

## Integración con supermercados online

## Integración con redes sociales

## Funcionalidad lista de la compra

## Agenda personal

## Multi-idioma

## Coach nutricional

## Análisis de sentimientos

## Integración con IBM IoT

## Notificaciones productos perecederos

## Cuadro de mando

* Productos consumidos
* Calorías consumidas

# Apéndice I: código

# Apéndice II: Guía de instalación

# Apéndice III: Manual de usuario

# Referencias

|  |  |
| --- | --- |
| [1] | Gartner, «The road to enterprise AI,» [En línea]. Available: https://www.gartner.com/imagesrv/media-products/pdf/rage\_frameworks/rage-frameworks-1-34JHQ0K.pdf. |
| [2] | TechEmergence, «AI and consumer technology,» [En línea]. Available: https://www.slideshare.net/TechEmergence/ai-founders-and-executives-predict-5year-trends-on-consumer-tech. |
| [3] | Minbowser, «Global Chatbot Trends Report – 2017,» [En línea]. Available: http://mindbowser.com/chatbot-market-survey-2017/. |
| [4] | S. team, «API Slack,» 2017. [En línea]. Available: https://api.slack.com/. |
| [5] | Z. Walchuk, «Easy Slack integration for Watson Conversation,» 2016. [En línea]. Available: https://www.ibm.com/blogs/watson/2016/10/easy-slack-integration-watson-conversation/. |
| [6] | M. Makai, «How to Build Your First Slack Bot with Python,» 2017. [En línea]. Available: https://www.fullstackpython.com/blog/build-first-slack-bot-python.html. |
| [7] | J. C. Fernandez, «Así funciona SiriKit,» [En línea]. Available: https://applecoding.com/guias/asi-funciona-sirikit. |

# **Índice de Conceptos**

conceptos 9

# **Glosario**

ISO: International Organization for Standardization 4

UNE: Una Norma Española 4

1. <https://www.ibm.com/watson/services/conversation> [↑](#footnote-ref-1)
2. <https://aws.amazon.com/lex> [↑](#footnote-ref-2)
3. <https://api.ai> [↑](#footnote-ref-3)
4. <https://www.luis.ai> [↑](#footnote-ref-4)
5. <https://wit.ai> [↑](#footnote-ref-5)
6. <https://github.com/snipsco/nlu-benchmark> [↑](#footnote-ref-6)
7. https://console.bluemix.net/docs/services/conversation/system-entities.html#system-entity-details [↑](#footnote-ref-7)
8. <http://docs.aws.amazon.com/lex/latest/dg/howitworks-builtins.html> [↑](#footnote-ref-8)
9. https://dialogflow.com/docs/entities#system-mapping [↑](#footnote-ref-9)
10. https://docs.microsoft.com/en-us/azure/cognitive-services/luis/pre-builtentities [↑](#footnote-ref-10)
11. <http://docs.aws.amazon.com/lex/latest/dg/versioning-aliases.html> [↑](#footnote-ref-11)
12. <https://docs.microsoft.com/en-us/azure/cognitive-services/luis/luis-how-to-manage-versions> [↑](#footnote-ref-12)
13. WebSocket es una tecnología que proporciona un canal de comunicación bidireccional y [full-duplex](https://es.wikipedia.org/wiki/Duplex_(telecomunicaciones)#Full-duplex) sobre un único [socket](https://es.wikipedia.org/wiki/Socket_de_Internet) [TCP](https://es.wikipedia.org/wiki/Transmission_Control_Protocol) [↑](#footnote-ref-13)
14. http://slackapi.github.io/python-slackclient/ [↑](#footnote-ref-14)