Universitat Oberta de Catalunya (UOC)

Máster Universitario en Ciencia de Datos (*Data Science*)

TRABAJO FINAL DE MÁSTER

Área: ¿Procesamiento de lenguaje natural?

Modelización de topic de llamadas en tiempo real

Borrador

Autor: Manuel E. Gómez Montero

Tutor UOC: Ana Valdivia Garcia

Tutor TE: Antonio Fernández Gallardo

Profesor: Jordi Casas?

Madrid, 14 de septiembre de 2019

Resumen

El documento que se presenta tiene como objetivo final mejorar la operatividad del *call-center* de una gran empresa, extrayendo mediante técnicas de *machine learning* la temática de las llamadas que se realizan al mismo y dando la posibilidad a la empresa de reaccionar en tiempo real, en función de la temática que se este tratando en cada momento.

Este nuevo sistema nos permitirá a partir de la trascripción de las llamadas al *call-center* de Telefónica España, descubrir en tiempo real la temática de las mismas. Esta modelización de *topics* se ha realizado utilizando métodos de procesamiento de lenguaje natural y *Deep Learning*. El sistema realiza la clasificación de las nuevas llamadas en tiempo real, permitiendo a los usuarios visualizar la evolución en la temática de las mismas y generar alertas en base a anomalías.

Palabras clave: "natural language processing", "sentiment analysis", "real time", "call center", "topic modeling", "deep learning"

Índice general

Capítulo 1

Introducción

Este primer capítulo del trabajo tiene como objetivo presentar, a grandes rasgos, la propuesta, los objetivos que pretendemos lograr, la motivación que nos ha llevado a abordar este proyecto y un repaso a las tareas que serán necesarias para la ejecución del mismo.

Por último, dedicaremos una sección que describa brevemente los diferentes apartados de los que constará el documento y el objetivo de cada uno.

1.1. Descripción general de la propuesta

En los últimos años, la explosión ingente en la generación de datos y el avance en las capacidades tecnológicas que nos permiten recolectar, almacenar y procesar los datos generados; han provocado que empecemos a analizar datos que hasta ahora no eran tenidos en cuenta. Un ejemplo de esto es el crecimiento en el tratamiento de datos no estructurados.

Dentro de los datos no estructurados, una de las fuentes de información con mayor potencial en todas las grandes empresas que prestan servicio al público general, son las llamadas que los clientes realizan a su *call-center*, ya que nos permiten obtener una idea de la percepción que los clientes tienen de nuestra empresa y de sus preocupaciones en cada momento.

La propuesta que pretendemos abordar en este trabajo consiste en extraer la temática de estas llamadas en el momento en el que son capturadas. Aunque ahora mismo esta captura se hace periódicamente pretendemos establecer una solución que nos permita el tratamiento de las mismas en tiempo real para que esta solución sea válida en un futuro próximo cuando se aumente la frecuencia de ingesta.

Esta extracción en tiempo real nos permitirá conocer cómo evolucionan los temas que tratan nuestros clientes cuando llaman a nuestro *call-center* y nos permitirá poder reaccionar inmediatamente ante una preocupación concreta.

1.2. Motivación

La motivación que nos ha llevado a acometer un proyecto de esta naturaleza viene originada por diferentes factores que están ligados tanto al negocio como a las capacidades técnicas disponibles en la empresa.

Por un lado, la capacidad de obtener la temática de las llamadas en tiempo real se presenta como una oportunidad inigualable de mejorar la operatividad de un *call-center* y por ende la satisfacción de los clientes, permitiéndonos entenderlos mejor y pudiendo reaccionar de una manera ágil a sus necesidades reales.

Desde el punto de vista técnico, también era el momento ideal para emprender este proyecto debido tanto a la disponibilidad periódica de transcripciones de las llamadas, que nos permiten ahorrarnos el paso de realizar un *Speech 2 Text* para obtener nuestro conjunto de datos; como al aumento de capacidades técnicas en la empresa que nos permitirán tanto entrenar nuestros modelos, como poder tratar y explotar los datos en tiempo real.

1.3. Objetivos

En este apartado definiremos los objetivos que se pretenden conseguir con este proyecto. Estos objetivos deben ser *SMART*, es decir:

- Específicos: Deben plantearse de una forma detallada y concreta.
- Medibles: Deben poder medirse con facilidad.
- Alcanzables: Deben ser objetivos realistas.
- Relevantes: Tienen que ser relevantes para la empresa y ofrecernos un beneficio claro.
- Tiempo: Estos objetivos tienen que tener un tiempo establecido.

Los objetivos principales que se pretenden conseguir con este proyecto son:

- **Construir un modelo que nos permita extraer la temática de las llamadas** a partir de su transcripción a texto. Este objetivo debemos alcanzarlo en la fase de modelado y podremos medir su éxito atendiendo al porcentaje de llamadas que podamos clasificar correctamente en un proceso de test. Se trata del objetivo principal del proyecto.
- Desarrollar un mecanismo que nos permita **extraer esta temática para nuevas llamadas en tiempo real**. De este modo tendremos un sistema vigente cuando la frecuencia en la recepción de las llamadas aumente. Este objetivo se deberá alcanzar en la fase de productivización.

- Disponer de una **visualización en tiempo cuasi real** para que pueda visualizarse la evolución de las temáticas a lo largo del tiempo. Este objetivo se deberá alcanzar en la fase de productivización.

- Proporcionar un **sistema de alertado** que nos permita detectar anomalías en el número de llamadas que se reciben de un determinado tema. Este objetivo se deberá alcanzar en la fase de productivización.

En las conclusiones de este proyecto se evaluará el éxito o fracaso del mismo en función del grado de cumplimiento de estos objetivos.

1.4. Tareas y planificación

El proyecto se llevará a cabo desde el 16 de Septiembre hasta el 20 de Febrero. Para poder abordar la ejecución del mismo se han extraído las siguientes tareas principales:

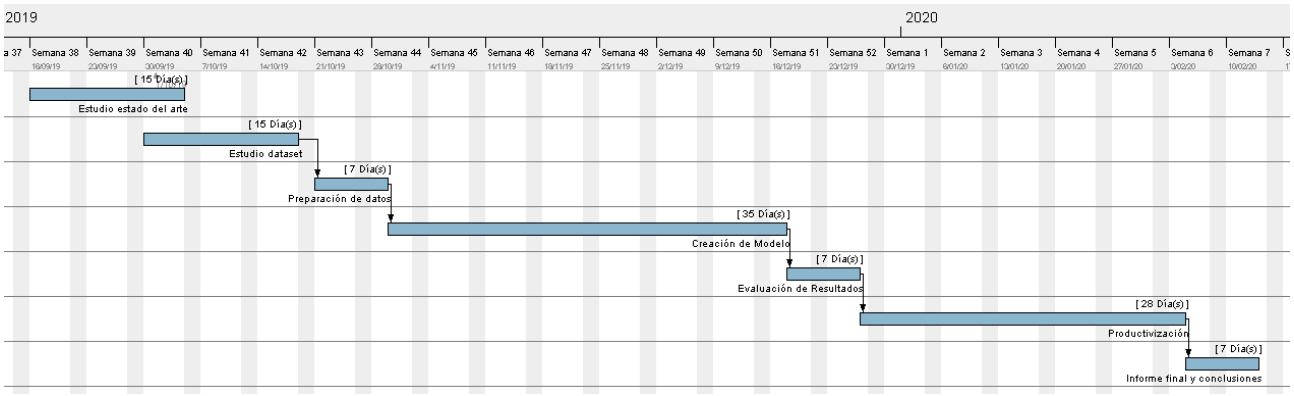

Figura 1.1: Diagrama de Gantt

- **Estudio estado del arte**: En esta fase se realizará una prospección para conocer el estado del arte en todos los puntos relacionados con el proyecto: Procesamiento del lenguaje natural, tecnologías de tratamiento de datos en tiempo real y Big Data.

- **Estudio del *dataset***: El propósito de esta tarea es entender el *dataset* y estudiar las posibilidades del mismo.

- **Preparación del *dataset***: Una vez realizado el estudio del *dataset* es necesario realizar labores de limpieza y transformación de los datos de modo que estos datos sean válidos para nuestro objetivo.

- **Creación del modelo**: En esta fase se procederá a la creación de un modelo capaz de obtener los temas de los que habla una determinada llamada. Este modelo será el *core* de nuestro proyecto.
- **Evaluación de resultados**: Una vez entrenado el modelo será necesario evaluar los resultados obtenidos para poder evaluar la bondad de nuestro modelo.
- **Productivización**: El trabajo no acaba con la creación de un buen modelo que nos permita extraer los temas de nuestras llamadas. Este modelo tendrá que ser puesto en producción y permitir al usuario final extraer los temas de las llamadas en tiempo real y darle la opción de crear alarmas basadas en la variación del número de eventos (llamadas) de un determinado tema.
- **Informe final y conclusiones**: Por último, una vez llevado a a producción nuestro modelo, se realizará un informe final donde, entre otros puntos, se evaluarán los resultados obtenidos y se extraerán conclusiones y pasos futuros.

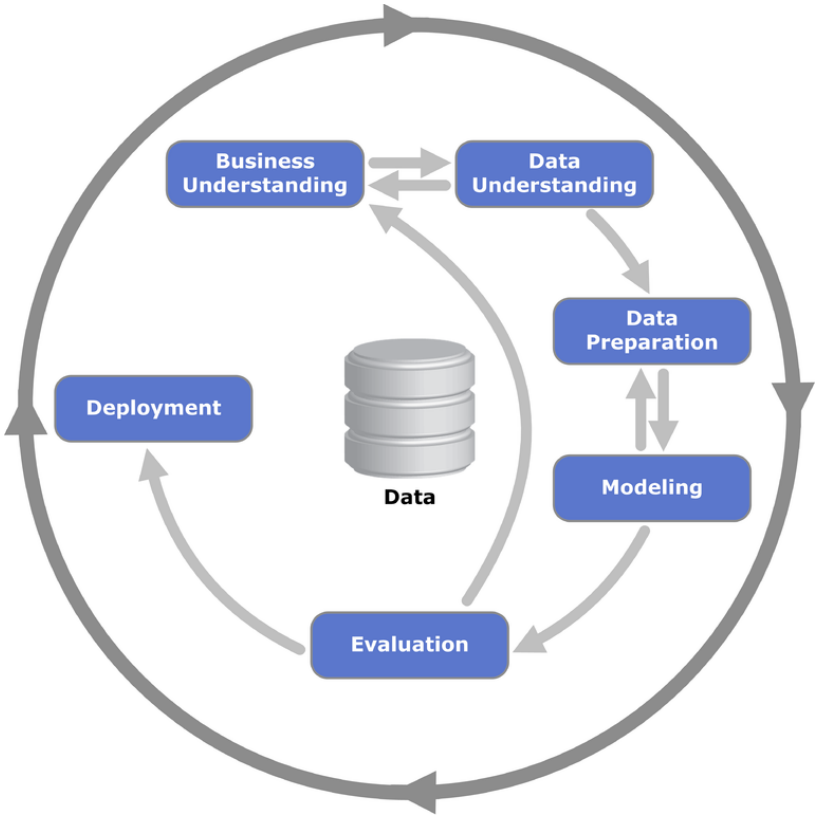

Figura 1.2: Fases del modelo CRISP-DM

Estas fases están basadas en el estándar ***CRISP-DM*** ([5]), añadiendo una última tarea para nuestro informe final, *CRISP-DM* nos proporciona una descripción del ciclo de vida de

los proyectos de minería de datos de un modo bastante similar al que se aplica en los modelos de ciclo de vida de desarrollo *software*.

En la figura 1.2 podemos ver una representación de este modelo y cómo representa el ciclo de vida de un proyecto de minería de datos. En la imagen podemos ver en primer lugar un círculo exterior que refleja la naturaleza cíclica de los proyectos de minería de datos, además vemos cómo la secuencia de tareas no es rígida, pudiendo saltar hacia adelante o atrás entre tareas. En la gráfica se representan mediante flechas las dependencias más importantes y usuales entre tareas.

En nuestro desarrollo usaremos este modelo, aunque en el diagrama de la figura 1.1 aparezca una secuencia de tareas más rígida, será usual, por ejemplo, el salto recíproco entre las fases de preparación de los datos y creación del modelo.

1.5. Estructura del documento

TODO Hacer repaso breve de los apartados del documento final.

Capítulo 2

Estado del Arte

El objetivo de este apartado es hacer un recorrido por el estado del arte relacionado con el proyecto, este recorrido lo enfocaremos desde tres puntos de vista diferentes:

- **Procesamiento de lenguaje natural**: En este apartado nos centraremos en el procesamiento del lenguaje natural y su evolución a lo largo del tiempo.
- ***Deep Learning* y aplicación al NLP**: En la segunda parte pondremos foco en el *Deep Learning*, sus ventajas y cómo se están aplicando estos métodos al procesamiento de lenguaje natural.
- ***Big Data* y *Fast Data***: Por último, haremos un repaso a la evolución del *Big Data* y cómo la tendencia actual es realizar el procesamiento en tiempo real mediante *Fast Data*.

2.1. Procesamiento de lenguaje natural

2.1.1. Historia

Para hablar de los orígenes del procesamiento del lenguaje natural (a partir de ahora NLP por sus siglas en inglés *Natural Lenguage Processing*) tal y como lo conocemos, tendríamos que remontarnos a los años 50, concretamente al artículo "*Computing Machinery and Intelligence*" escrito por Alan Turing [15]. En este artículo aparece el NLP dentro del campo de la inteligencia artificial y se presenta por primera vez el conocido "Test de Turing". Este test convirtió la pregunta abstracta de "¿Son capaces de pensar las máquinas?" en un juego llamado: "*The Imitation Game*". El juego propuesto inicialmente, de forma muy resumida, consiste en ver si una persona (interrogador) interrogando a dos personas (un hombre y una mujer), era capaz de descubrir el sexo de cada una; la modificación del mismo sustituye las dos personas de distinto sexo por una persona y una máquina y el interrogador debe ser capaz de descubrir si

las preguntas están siendo respondidas por un humano o una máquina. En el caso de que no sepa discernir, la computadora gana la partida. Podemos encontrar más información al respecto en el libro [16].

A partir de los avances de Turing y hasta los 80 el crecimiento en el campo del NLP se produjo principalmente con la creación de complejos sistemas basados en reglas escritas a mano. Fue en esta década cuando empezamos a vivir la incorporación de algoritmos de *machine learning* enfocados al procesamiento del lenguaje natural. Este hecho se vio motivado principalmente por el increíble avance en la capacidad de cómputo, ya predicho por la ley de Moore, y por la aplicación de teorías ya existentes como los trabajos de Chomsky.

Desde el comienzo de la aplicación de modelos de machine learning, y de nuevo motivados por el crecimiento de la capacidad computacional de los sistemas actuales, se ha pasado de utilizar árboles de decisión, que creaban de manera automática reglas similares a las que se venían creando manualmente, a los modelos de *deep learning* que están en auge en la última década.

2.1.2. Aplicaciones

En el apartado anterior hicimos referencia a "*The Imitation Game*" como inicio de lo que hoy conocemos como procesamiento de lenguaje natural, sin embargo, las aplicaciones en este campo han crecido de forma vertiginosa en estos 70 años, principalmente en las últimas décadas. Hoy en día, si tuviéramos que contestar a la pregunta: "¿son capaces de pensar las máquinas?", implicaría algo más que superar el test de Turing. Mirando a nuestro alrededor nos encontraríamos con asistentes de voz como Alexa o Siri que no solo contestan a nuestras preguntas, si no que realizan un trabajo de pasar nuestra voz a texto (*Speech to Text*) y de nuevo el texto resultante a voz (*Text to Speech*). Nos encontraríamos también con sistemas capaces de realizar traducciones simultaneas, otros capaces de autocompletar textos, de identificar preguntas y respuestas, de clasificar textos de acuerdo a temas o autores, incluso de analizar sentimientos positivos o negativos teniendo como entrada un texto u opinión...

Según [10] todos estos problemas tan diversos podríamos clasificarlos según en el punto del análisis que nos centremos:

- **Análisis de palabras**: En este tipo de problemas se pone foco en las palabras, como pueden ser "perro", "hablar", "piedra" y necesitamos decir algo sobre ellas. Por ejemplo: "¿estamos hablando de un ser vivo?", "¿a qué lenguaje pertenece?", "¿cuáles son sus sinónimos o antónimos?". Actualmente este tipo de problemas son menos frecuentes, ya que normalmente no pretendemos analizar palabras aisladas sino que es preferible basarse en un contexto.

- **Análisis de textos**: En este tipo de problemas no trabajamos solo con palabras aisladas, sino que disponemos de una pieza de texto que puede ser una frase, un párrafo o un documento completo y tenemos que decir algo sobre él. Por ejemplo: "¿se trata de spam?", "¿qué tipo de texto es?", "¿el tono es positivo o negativo?", "¿quién es su autor?".Este tipo de problemas son muy comunes y nos vamos a referir a ellos como **problemas de clasificación de documentos**.

- **Análisis de textos pareados**: En esta clase de análisis disponemos de dos textos (también podrían ser palabras aisladas) y tenemos que decir algo sobre ellos. Por ejemplo, "¿los textos son del mismo autor?", "¿son pregunta y respuesta?", "¿son sinónimos?" (para el caso de palabras aisladas).

- **Análisis de palabras en contexto**: En estos casos de uso, a diferencia del primer análisis que trataba unicamente con palabras aisladas, tenemos que clasificar una palabra en particular en función del contexto en el que se encuentra.

- **Análisis de relación entre palabras**: Este último tipo de análisis tiene como objetivo deducir la relación entre dos palabras existentes en un documento.

Dependiendo del problema que queramos abordar usaremos un tipo de características del lenguaje u otro, por ejemplo, es usual que si estamos analizando palabras aisladas nos centremos en las letras de una palabra, sus prefijos o sufijos, su longitud, la información léxica extraída de diccionarios como *WordNet* [8]... En cambio, si estamos trabajando con texto, lo normal es que nos fijemos en otros conceptos estadísticos como el histograma de las palabras dentro del texto, ratio de palabras cortas vs largas, número de veces que aparece una palabra en un texto comparado con el resto de textos...

El proyecto que se presenta en este documento está centrado en el análisis de textos, concretamente en extraer los temas de un documento (o llamada). Este tipo de problemas se conoce como modelización de *topics*.

En el siguiente punto de este apartado nos centraremos en algunos modelos y avances en este área que puedan servirnos de apoyo para nuestro proyecto.

2.1.3. Modelización de topics

Técnicamente hablando, la modelización de topics hace referencia a un grupo de algoritmos de *machine learning* que infieren la estructura latente existente en un grupo de documentos.

Aunque la mayoría de los algoritmos de modelización son no supervisados, al igual que los algoritmos tradicionales de *clustering*, existen también algunas variantes supervisadas que necesitan disponer de documentos etiquetados.

Quizás el algoritmo más conocido para la modelización de topics sea el *Latent Dirichlet Allocation* (normalmente conocido por su acrónimo, LDA). LDA fué presentado en 2003 en el artículo [4]. Este algoritmo no supervisado asume que cada documento es distribución probabilística de *topics* y cada *topic*, a su vez, es una distribución de palabras del documento. LDA usa una aproximación llamada "*bag of words*", en la que cada documento es tratado como un vector con el conteo de las palabras que aparecen en el mismo. La principal característica de LDA es que la colección de documentos comparten los mismos topics, pero cada documento contiene esos topics en una proporción diferente.

A partir de LDA surgieron numerosas variantes que repasaremos de forma breve, por ejemplo, en el mismo año de la creación de LDA y también presentado por los mismos autores en [1], surgió una **variante jerárquica** que permitía representar los *topics* jerarquicamente. En 2006 en [3] se desarrolla un modelo LDA dinámico denominado DTM (*Dinamic Topic Model*), en el que se introduce la variable temporal y los *topics* pueden ir cambiando a lo largo del tiempo. En el artículo [2] nos encontramos con otra variante de LDA llamada CTM (*Correlated topic model*) que nos permite encontrar correlaciones entre *topics*, ya que algunos temas es probable que sean más similares entre sí. Por último, nos encontramos con una variante de LDA denominada ATM (*Author-Topic Model*) propuesta por Michal Rosen-Zvi en su artículo [14] y desarrollada por el mismo en 2010, en la que los documentos son una distribución probabilística tanto de autores como de *topics*.

Podemos encontrar un resumen más completo del estado del arte en cuanto a la modelización de *topics* en el artículo [12].

2.2. Deep Learning y aplicación al NLP

El objetivo de esta sesión es entender el concepto de *Deep Learning* y analizar el estado del arte del *Deep Learning* aplicado al procesamiento del lenguaje natural. Para poder entender el *Deep Learning* es conveniente entender los modelos de aprendizaje supervisados y saber qué provoca su aparición y popularidad de los últimos años. Posteriormente nos centraremos en los fundamentos del *Deep Learning* para finalizar con las aplicaciones actuales en el ámbito del procesamiento del lenguaje natural y que nos sirvan de apoyo para la ejecución del proyecto.

2.2.1. Aprendizaje supervisado

El aprendizaje supervisado consiste en aprender una función a través de un conjunto de datos, llamados de entrenamiento, mediante la cual podamos obtener una salida a partir de una determinada entrada. Se espera que esta función, una vez realizado el entrenamiento, sea

capaz de producir una salida correcta incluso para datos nunca vistos. Es muy habitual el uso de estos tipos de algoritmos para casos de clasificación y/o predicción.

Buscar entre todas las posibles infinitas funciones posibles para encontrar la función que mejor se adapte a nuestro conjunto de datos es un trabajo inviable, es por ello que normalmente se realiza la búsqueda entre un conjunto de funciones limitadas. En un primer lugar, y hasta hace aproximadamente una década, los modelos más populares de aprendizaje supervisado fueron los modelos lineales, provenientes del mundo de la estadística, estos modelos son fáciles de entrenar, fáciles de interpretar y muy efectivos en la práctica.

A partir de entonces, y motivado en parte por el aumento en las capacidades de cómputo, surgen otros modelos como las máquinas de vectores de soportes (*Support Vector Machines*, SVMs) o las redes neuronales, en las que nos centraremos en el siguiente apartado.

2.2.2. Deep Learning

Dentro del *Machine Learning* y usualmente relacionado con el aprendizaje supervisado, nos encontramos con un sub-campo denominado **Deep Learning** que utiliza las redes neuronales para la creación de modelos. Como su nombre indica las redes neuronales consisten en unidades de cómputo llamadas neuronas que están interconectadas entre sí. Una neurona es una unidad de cómputo que posee múltiples entradas y una salida, esta neurona multiplica cada entrada por un peso para posteriormente realizar una suma y, por último, aplicar una función de salida no lineal. Si los pesos se establecen correctamente y tenemos un número suficiente de neuronas, una red neuronal puede aproximar a un conjunto muy amplio de funciones matemáticas.

En las redes neuronales, las neuronas suelen organizarse por capas que se encuentran conectadas entre sí. Mientras más capas tengamos más características podremos extraer de nuestros datos de entrada y podremos aproximar un mayor número de funciones (sin perder de vista el sobrentrenamiento). Hablamos que una red es profunda cuando contiene un gran número de capas, por ello el término de *Deep Learning*.

2.2.3. Redes neuronales en NLP

Es usual, en el ámbito del reconocimiento de imágenes, utilizar información acerca de la dimensionalidad de las mismas. Este tipo de información nos permite extraer características teniendo en cuenta los píxeles vecinos. Tradicionalmente, en el ámbito del procesamiento del lenguaje natural, esto no se ha llevado a cabo debido a que cada palabra (o n-grama) se trataba como una entidad aislada.

En cambio, existe otro método de representar las palabras en el lenguaje natural que sí es capaz de captar la "dimensionalidad" de una forma similar a como lo realizamos en las

imágenes. Este modo deja de tratar la palabra como un ente aislado y es capaz de captar el significado de la misma, esta representación se denomina distribuida y consiste en convertir las palabras en vectores en los que cada dimensión capte características diferentes de las palabras. Este tipo de representaciones dará lugar a vectores similares para palabras semánticamente parecidas.

Una de las soluciones más populares que nos permiten convertir una palabra a un vector (*word2vec*) que contenga información de la palabra en función del contexto se detallan en el artículo [13] . Aquí se presentaron dos modelos llamados Skip-Gram y CBOW. Estos modelos utilizan redes neuronales para predecir la siguiente palabra o para predecir una palabra en función de su contexto, el vector que se utiliza posteriormente para representar la palabra es el vector de pesos de la capa oculta.

TODO comentar Arquitecturas, Recurrentes, convolucionales... BERT ... En función de cómo orientemos el modelo.

2.3. BigData y Fast Data

El primer uso del término *Big Data* se da en un artículo de Michael Cox y David Ellsworth de la NASA publicado en 1997 ([6]), donde hacen referencia a la dificultad de procesar grandes volúmenes de datos con los métodos de la época. Sin embargo, fue en 2001 cuando encontramos la definición más conocida y aceptada de *Big Data* hecha por el analista Laney Douglas en su artículo "3D Data Management: Controlling Data Volume, Velocity y Variety"([11]) en el que se hacía referencia a las ya "famosas" tres *V*s:

- **V**olumen: Cada vez los volúmenes de datos son mayores.
- **V**elocidad: Es cada vez mayor la velocidad con la que se generan los datos.
- **V**ariedad: Dejamos de tener únicamente datos completamente estructurados para trabajar con datos no estructurados y/o semi-estructurados.

Google, como es obvio, también se enfrentó a un importante problema a la hora de procesar la ingente cantidad de datos que generaba día a día y que no podían ser procesados de manera eficiente con el *software* existente, es por ello que en el año 2003 presenta en [9] su *"Google File System"* (GFS) y un año después *Map Reduce* [7], estas dos capas de almacenamiento y procesamiento distribuido dieron lugar al nacimiento de lo que hoy conocemos como ***Big Data***.

Sin embargo, estas aportaciones no empezaron a tomar una repercusión relevante fuera de Google hasta el nacimiento del *framework* Hadoop en 2006, un ecosistema con una gran cantidad de servicios pero cuya base fue Map Reduce y HDFS (basado en GFS). La complejidad del

ecosistema *Hadoop* hizo que éste no empezara a aparecer en las mayoría de las empresas hasta la creación de la compañía *Cloudera* en 2009, que empezó a empaquetar los diferentes componentes del ecosistema *Hadoop*, ofreciendo distribuciones estables y soporte para sus clientes.

Durante estos 10 años la popularidad de *Hadoop* ha crecido exponencialmente y junto con las BBDD NoSQL, nacidas también a partir de Google con su BigTable, forman lo que hoy conocemos como Big Data.

El auge del **Big Data** ha llevado a algunas empresas a tener verdadera obsesión por el almacenamiento de todos los datos de sus clientes y las operaciones realizadas, creando inmensos *datalakes* donde tener enormes históricos de todos sus datos, este "síndrome de Diógenes digital" creado por falsas expectativas, por la imposibilidad de extraer valor de los datos o por la dimensión cambiante de las empresas actuales, en la que los datos de años atrás pueden no ser relevantes en el presente, es uno de los posibles motivos por lo que el tratamiento de los datos esta cambiando. Otro de los motivos para el cambio de rumbo del *Big Data* está relacionado con la *V* de Velocidad, hoy en día no solo es importante la capacidad de ingestar rápidamente los datos, sino la capacidad de poder procesar y obtener decisiones o actuar en tiempo real a partir de los datos, aportando valor al negocio. En este escenario se vuelve más importante la velocidad que el volumen de datos, esto es lo que se denomina *Fast Data*.

Dentro del *Fast Data* es habitual el uso de BBDD *in-memory*, de buses de eventos y de tecnologías de procesamiento capaces de procesar los eventos en tiempo real. Como veremos posteriormente al desarrollar nuestra arquitectura, el *Fast Data* será una parte fundamental en nuestro proyecto en el que tendremos que clasificar las llamadas en tiempo real y tomar decisiones (o alarmar) en función de las mismas.

Capítulo 3

Arquitectura y tecnologías

El objetivo de este capítulo es tener un diseño esquemático de la solución que se plantée, este diseño será abordado en primer lugar desde un punto de vista lógico, de acuerdo a las necesidades del proyecto, posteriormente aterrizaremos esta arquitectura con tecnologías concretas que nos ayuden a conseguir el objetivo deseado.

3.1. Arquitectura

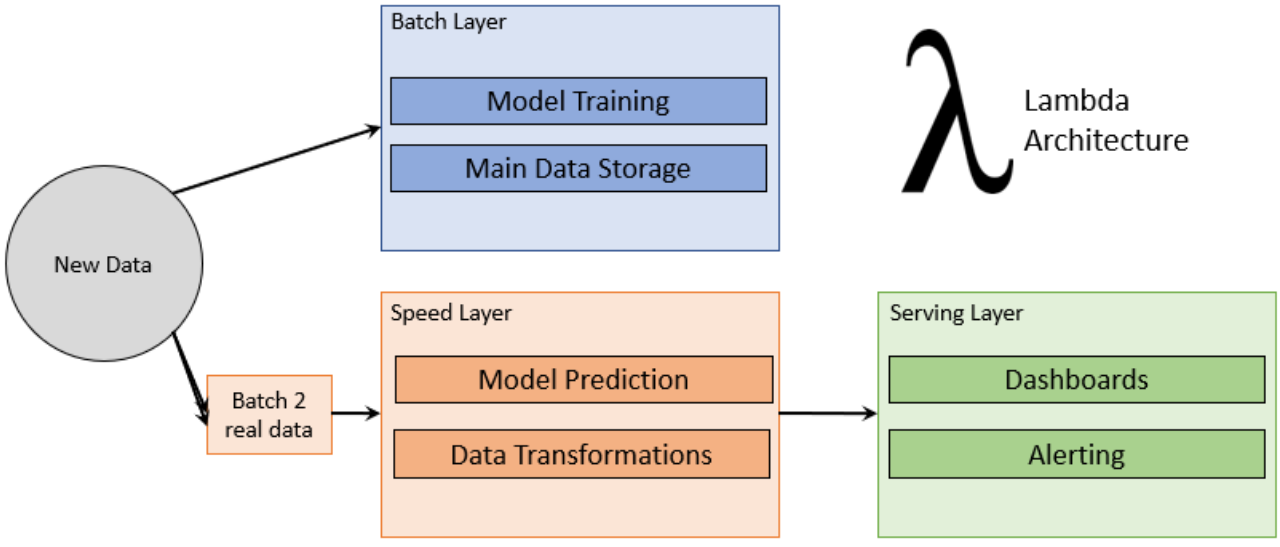

Figura 3.1: Arquitectura Lambda propuesta

En este apartado definiremos la arquitectura desde un punto de vista lógico, esta arquitectura debe responder a los objetivos propuestos para cumplir con las necesidades de negocio

existentes.

En líneas generales, podemos ver la arquitectura de nuestro sistema como una arquitectura *Lambda* en la que disponemos de una capa *batch*, una capa rápida o *real-time* y una última capa de servicio.

La capa batch será la encargada de entrenar el modelo a través de los datos de las llamadas, la capa rápida obtendrá los *topics* de las llamadas en tiempo real usando el modelo previamente entrenado y por último la capa de servicio será la encargada de mostrar estos datos al usuario mediante cuadros de mando.

En la figura 3.1 podemos ver un esquema general de nuestra arquitectura. A continuación definimos con más detalle cada una de las capas del modelo.

3.1.1. Capa Batch

El *core* del proyecto que abordamos es el modelo, encargado de extraer los *topics* de las transcripciones de llamadas al servicio de atención al cliente. Este modelo debe entrenarse usando un histórico suficientemente amplio.

El modelo es un elemento vivo en nuestra arquitectura y, además de por posibles mejoras en los hiperparámetros o por la tecnología, debe re-entrenarse conforme se vayan recibiendo datos nuevos en el histórico, ya que es lógico pensar que la temática de las consultas variarán a lo largo del tiempo debido por ejemplo al lanzamiento de nuevos productos.

3.1.2. Capa Real-Time

La *speed layer* de nuestro proyecto será la encargada de recibir los datos en tiempo real, las llamadas serán publicadas en un bus de eventos y estos eventos serán consumidos por una capa de procesamiento que será la encargada de aplicar el modelo entrenado en la capa *batch* a los nuevos datos. Los *topics* resultantes de cada llamada serán publicados de nuevo en este bus para poder ser ingestados posteriormente a una BBDD NO-SQL, que será la encargada de proporcionar la información a la capa de servicio.

Aprovecharemos también las características de la BBDD NoSQL para, mediante un módulo de *Machine Learning*, detectar anomalías en series temporales y poder alarmar en el caso de que un *topic* concreto se dispare en algún momento.

Debido a la situación actual, las llamadas no se ingestan en *real-time* si no que se procesan en mini *batches* cada cierto tiempo. Es muy probable que este escenario cambie en el futuro por lo que se construirá un elemento de entrada a la capa rápida que transformará el mini-batch en eventos conforme vayan ejecutándose (este elemento puede observarse en la figura 3.1 como *Batch 2 real data*). Esta pieza será suprimida una vez las llamadas sean recibidas en tiempo

real.

3.1.3. Capa de Servicio

Una vez almacenados los datos en la BBDD debemos construir un frontal que muestre al usuario el número de llamadas analizadas, el modelo utilizado y principalmente la evolución de los *topics* a lo largo del tiempo.

Lo ideal es que esta capa de servicio se vaya actualizando en tiempo cuasi-real y permita a los diferentes usuarios realizar cuadros de mando personalizados según sus necesidades.

Otro requisito esencial en este tipo de proyectos es que la información este disponible vía API-REST para poder ser consumida por terceros.

3.2. Integración y Despliegue Continuos

Como ya hemos visto al hablar del entrenamiento del modelo, el desarrollo de este tipo de proyectos no tiene un principio y un final, si no que se trata de un proceso cíclico en el que por necesidades del negocio, por cambio en los datos o por cambio en la tecnologías, será necesario añadir mejoras o modificaciones en nuestro desarrollo.

Por estos motivos es conveniente definir mecanismos que nos permitan, tras cada cambio efectuado, poder realizar las pruebas necesarias y desplegar estos cambios de una manera totalmente automatizada y sin intervención del usuario.

Aunque este apartado queda fuera de la implementación inicial del proyecto es imprescindible llevarlo a cabo para que este sea sostenible a lo largo del tiempo.

TODO (Posible ampliar con apuntes sobre el ciclo de vida de los datos y contar beneficios de un despliegue continuo)

3.3. Tecnologías

Al igual que la arquitectura descrita anteriormente era la encargada de responder a las necesidades de negocio, las tecnologías descritas en este apartado nos darán las piezas necesarias para poder construir esa arquitectura y dar respuesta a nuestro caso de uso.

En el proceso de selección de las tecnologías, no solo se ha tenido en cuenta la idoneidad de las mismas para el caso de uso, si no que se ha valorado también la experiencia en la misma y la disponibilidad dentro del entorno de trabajo. Esto puede provocar que en algunos casos aunque la tecnología se adapte al caso de uso, puedan existir otras soluciones más óptimas cuyo uso era menos viable dado los plazos de ejecución del proyecto.

A continuación enumeraremos las tecnologías agrupadas en las diferentes capas que hemos comentado en el apartado de arquitectura, además añadiremos las tecnologías que se usarán para la integración y despliegue continuo.

3.3.1. Capa batch

- **Spark**: Es un framework de computación en clúster. Este *framework* se encuentra desplegado sobre un clúster Hadoop, específicamente una distribución HortonWorks, y posee librerías específicas para Machine Learning como MLLIB.

- **Tensorflow** sobre GPUs: Para el entrenamiento de los modelos también disponemos de un cluster con GPUs sobre el que podremos correr código usando la librería de Google Tensorflow para entrenar nuestros modelos.

3.3.2. Capa Real-Time

- **Kafka**: Es el *core* de la capa rápida, se trata de un bus de evento distribuido a través del cual se realizara la ingesta o publicación de los eventos (llamadas). Las diferentes capas de procesamiento que requieran estos eventos se suscribirán a este Bus.

- **Microservicios**: En nuestra capa Real Time construiremos diferentes microservicios en la capa de procesamiento, estos microservicios serán por ejemplo los encargados de devolver los *topics* de cada llamada a partir de una llamada API REST. Estos microservicios correran sobre contenedores en una plataforma Openshift.

- **Kafka Stream y KSQL**: A la hora de procesar la información en eventos ingestada en nuestro Bus Kafka disponemos de dos herramientas muy potentes que son Kafka Stream y KSQL. El primero consiste en una serie de librerías que nos permiten construir aplicaciones y microservicios cuyo origen y destino sean un Bus Kafka. KSQL en cambio es un motor SQL aplicable a eventos que nos llegan mediante *streaming*.

- **Logstash**: Una vez hayamos extraído los *topics* correspondientes a cada llamada o evento, tendremos que ingestar esta información en nuestra BBDD, que en este caso será Elasticsearch. Logstash nos permitirá leer de Kafka, realizar las transformaciones necesarias y volcar la información resultante en Elasticsearch.

- **Elasticsearch**: Aunque no se trata en el sentido más estricto de una BBDD No-SQL, si no de un motor de búsqueda, Elasticsearch nos permite almacenar la información en forma de documentos json y realizar consultas y agregaciones sobre cualquier campo.

Entre las características que podemos aprovechar de Elasticsearch para nuestro objetivo están:

- Ingesta en tiempo casi real.
- Consulta en tiempo casi real.
- Disponibilidad de mecanismos de ingesta (Logstash) y consulta (kibana).
- Posibilidad de crear alarmas en base a consultas.
- Posibilidad de crear *jobs* de Machine Learning que detecten anomalías en series temporales.

3.3.3. Capa Servicio

- **Elasticsearch API REST**: Toda la información almacenada previamente en Elasticsearch puede ser accedida a través de su API REST por lo que será nuestro método de publicación de la información.
- **Kibana**: Será el frontal donde los usuarios podrán consultar sus diferentes cuadros de mando y construir nuevos de acuerdo a sus necesidades. También, gracias al modulo de *machine learning* de Elasticsearch, los usuarios podrán crear *jobs* de *machine learning* para detectar anomalías en los temas tratados y generar las alertas necesarias.

3.3.4. Integración y Despliegue Continuo

Aunque la implementación de esta parte se escapa de los plazos del proyecto, las tecnologías que se usarán para llevarla a cabo serán.

- **BitBucket**: Será el repositorio usado para almacenar las nuevas versiones de nuestro *software* de manera que podamos tener un control de versiones.
- **Jenkins**: Es un servidor de integración continua *open source* que mediante la creación de tareas nos ayudará a realizar el *build* de nuestro software realizando de manera automática las pruebas necesarias.

Capítulo 4

Conjunto de datos

Bibliografía

[1] David M. Blei, Michael I. Jordan, Thomas L. Griffiths, and Joshua B. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, pages 17–24, Cambridge, MA, USA, 2003. MIT Press.

[2] David M. Blei and John D. Lafferty. Correlated topic models. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, NIPS'05, pages 147–154, Cambridge, MA, USA, 2005. MIT Press.

[3] David M. Blei and John D. Lafferty. Dynamic topic models. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 113–120, New York, NY, USA, 2006. ACM.

[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[5] Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rudiger Wirth. Crisp-dm 1.0 step-by-step data mining guide. Technical report, The CRISP-DM consortium, August 2000.

[6] Michael Cox and David Ellsworth. Application-controlled demand paging for out-of-core visualization. In *Proceedings of the 8th Conference on Visualization '97*, VIS '97, pages 235–ff., Los Alamitos, CA, USA, 1997. IEEE Computer Society Press.

[7] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation - Volume 6*, OSDI'04, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[8] Christine Fellbaum. *WordNet: an electronic lexical database*. MIT Press, 1998.

[9] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. *SIGOPS Oper. Syst. Rev.*, 37(5):29–43, October 2003.

[10] Yoav Goldberg. *Neural network methods in natural language processing*. Morgan Claypool publishers, 2017.

[11] Douglas Laney. 3D data management: Controlling data volume, velocity, and variety. Technical report, META Group, February 2001.

[12] A. S. M. Ashique Mahmood. *Literature Survey on Topic Modeling*. 2013.

[13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[14] Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. The author-topic model for authors and documents. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, pages 487–494, Arlington, Virginia, United States, 2004. AUAI Press.

[15] A. M. Turing. *Computing machinery and intelligence*. Blackwell for the Mind Association, 1950.

[16] Kevin Warwick and Huma Shah. *Turings imitation game: conversations with the unknown*. Cambridge Univeristy Press, 2016.