MÁSTER UNIVERSITARIO EN BIG DATA

TRABAJO FIN DE MASTER

SPEECH ANALYTICS:

Procesamiento del lenguaje natural aplicado a la relación de la empresa y sus clientes

Autor: Alberto España Carrera

Director: Antonio Fernández Gallardo

Madrid

Mayo de 2025

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Speech Analytics: Procesamiento del lenguaje natural aplicado a la relación de la empresa y sus clientes

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2024/2025 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Alberto España Carrera Fecha: 10/05/2025

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Antonio Fernandez Gallardo

Firmado digitalmente por Antonio Fernandez Gallardo
Fecha: 2025.05.12 18:08:36 +02'00'

Fdo.: Antonio Fernández Gallardo Fecha: 10/05/2025

SPEECH ANALYTICS: PROCESAMIENTO DEL LENGUAJE NATURAL APLICADO A LA RELACIÓN DE LA EMPRESA Y SUS CLIENTES

Autor: España Carrera, Alberto.
Director: Fernández Gallardo, Antonio.
Entidad Colaboradora: Telefónica España

RESUMEN DEL PROYECTO

Este proyecto tiene como objetivo investigar cómo los clientes de Telefónica España se comunican con la compañía a través del canal telefónico 1004. Para lograrlo, se analizan diferentes modelos que utilizan técnicas avanzadas de deep learning enfocadas en el procesamiento de texto. Un paso fundamental en el proceso consiste en transformar las grabaciones de audio en texto, tarea que se aborda mediante una comparación de varias soluciones de transcripción automatizada.

Palabras clave: Inteligencia Artificial, Scraping, speech-to-text, deep learning, natural language processing (NLP), GPUs, machine learning.

1. Introducción

El propósito de este proyecto es evaluar diferentes algoritmos de Inteligencia Artificial que permitan extraer valor de las comunicaciones telefónicas entre Telefónica España y los usuarios, a través del canal telefónico.

2. Definición del proyecto

El proyecto se estructura en tres fases claramente diferenciadas:

- Obtener el audio con la conversación entre empresa y cliente, para lo que es necesario emplear herramientas de web scraping.
- Tratar dicho audio, en particular convirtiéndolo a texto
- Desarrollo y evaluación de modelos de machine learning para los distintos casos de uso.

Figura 1. Fases del proyecto Speech Analytics

3. Descripción de modelos y casos de uso aplicables

El proyecto se enfoca en identificar clientes insatisfechos a partir del análisis de las interacciones entre la empresa y sus usuarios, con el fin de implementar estrategias que reduzcan el riesgo de abandono y, por ende, disminuyan la tasa de churn. Aunque existen múltiples aplicaciones derivadas de este tipo de análisis, esta es la prioridad central.

Inicialmente, se utiliza un modelo construido con fastText (https://fasttext.cc/). Posteriormente, se emplean arquitecturas avanzadas que combinan redes convolucionales con redes LSTM (Long Short-Term Memory) y GRU (Gated Recurrent Unit), implementadas mediante la biblioteca Keras (https://keras.io/). A continuación, se desarrolla un modelo que utiliza BERT (Bidirectional Encoder Representations from Transformer), una herramienta de Google (1). Finalmente, se evalúan alternativas a esta última herramienta que podrían aportar mejoras en resultados, en particular en cuanto a su adaptación a casos de uso en lengua castellana.

4. Resultados y conclusiones

En la fase inicial del proyecto, se trabajó en la automatización del proceso de descarga masiva de audios. Paralelamente, se llevó a cabo la transcripción manual de un conjunto de llamadas, lo que permitió comparar el desempeño de cuatro servicios de transcripción. Aunque uno de ellos presentó resultados significativamente inferiores, no fue sencillo determinar de manera objetiva cuál de las otras tres opciones era la más adecuada. Además, la selección final de la herramienta trasciende las competencias del equipo de trabajo.

Respecto al rendimiento de los modelos, el uso de GPUs mostró beneficios importantes. Sin embargo, dado que estas unidades no siempre están disponibles, se diseñó un modelo optimizado para su funcionamiento en CPUs. A pesar de ello, las técnicas más avanzadas aún dependen del uso de unidades gráficas para su implementación.

En las etapas siguientes del proyecto, se recomienda incorporar técnicas de deep learning aunque su ajuste puede implicar un elevado costo computacional. Además, considerando que en deep learning el tamaño del conjunto de datos es un factor crucial, se propone aumentar la cantidad de llamadas descargadas y transcritas para mejorar el desempeño de los modelos.

Índice de la memoria

Índice de figuras

Índice de tablas

Índice de ecuaciones

Capítulo 1. INTRODUCCIÓN

El procesamiento del lenguaje natural, conocido como NLP (Natural Language Processing) en inglés, es una rama de la informática y la inteligencia artificial enfocada en analizar y comprender el lenguaje humano. Su principal objetivo es desarrollar sistemas que permitan a los ordenadores interpretar, procesar y generar lenguaje de manera similar a como lo hacen las personas. En los últimos años, el interés por esta disciplina ha crecido exponencialmente, impulsado por avances en modelos como GPT y BERT, y por la expansión de aplicaciones prácticas como los asistentes virtuales (Siri, Alexa, Google Assistant, Aura, entre otros), los chatbots inteligentes para atención al cliente, y herramientas más avanzadas de traducción automática y generación de texto.

Una de las aplicaciones más relevantes del NLP es el análisis de sentimiento, que permite procesar grandes volúmenes de texto para identificar la actitud, emoción o posición de los interlocutores frente a un tema específico. Estas técnicas han cobrado especial relevancia en áreas como la monitorización de redes sociales, el análisis de reseñas de productos y la mejora de la experiencia del usuario en diferentes servicios digitales.

1.1 MOTIVACIÓN DEL PROYECTO

El objetivo de Telefónica España (**a lo largo del documento se referirá a la compañía como "Telefónica" o "Telefónica España" indistintamente**) es la aplicación de técnicas de procesamiento del lenguaje natural para conocer con más detalle la interlocución de la compañía con clientes a través del canal 1004. El primer paso del proceso es transformar las grabaciones de las interacciones con los usuarios en texto, procesarlo adecuadamente y construir modelos de machine learning que se adapten a diversos escenarios prácticos. Estos modelos hacen posible entender mejor las demandas de los clientes, detectar áreas que requieren optimización y, en última instancia, implementar medidas que beneficien a la

organización. Esto se logra aprovechando datos internos de la empresa que, hasta ahora, no se habían explotado al máximo para generar valor estratégico.

1.2 FASES DEL PROYECTO

Como en cualquier proyecto de analítica de datos, el primer paso consiste en recopilar los datos y procesarlos para obtener la información en el formato necesario. En este caso, las grabaciones de las llamadas realizadas a través del canal 1004 se obtienen mediante un proceso de scraping. Posteriormente, se contacta con el cliente mediante una llamada automatizada días después de la interacción, solicitándole que evalúe la calidad del servicio recibido. Esto permite a Telefónica asignar un nivel de "satisfacción" a cada interacción, siempre que el cliente responda a la encuesta. Además, la compañía cuenta con información adicional vinculada a cada llamada, como si esta resultó en una contratación o no.

El siguiente paso en el flujo de trabajo consiste en transcribir los audios obtenidos de la manera más precisa posible. Para abordar esta fase, se consideraron cuatro herramientas diferentes, realizando una comparativa detallada entre ellas con el fin de elegir la más adecuada. Como se verá, el texto generado por cada sistema de transcripción necesita ser procesado para facilitar una comparación justa y uniforme. Para establecer un patrón de referencia y poder evaluar con precisión cada uno de los sistemas de transcripción, se transcribieron manualmente 67 llamadas. Debido a que varios miembros del equipo participaron en esta tarea, se creó un conjunto de normas específicas para manejar cuestiones como acentos, expresiones idiomáticas y la transcripción de números, lo que aseguraba coherencia en el proceso.

Una vez que se dispone de las transcripciones junto con sus metadatos asociados, se pasa a la fase de desarrollo de los modelos de machine learning. Aunque los detalles técnicos serán profundizados más adelante, los casos de uso que se abordan principalmente corresponden a análisis de sentimiento. Dado que este tipo de análisis implica interpretar matices complejos de emociones y opiniones, las técnicas de deep learning resultan esenciales. Por lo tanto, se opta por utilizar GPUs tanto en la nube, aprovechando la flexibilidad de

escalabilidad que ofrece, como en la infraestructura local durante las etapas finales del proyecto, lo que permite optimizar el rendimiento de los modelos en función de las necesidades específicas del proceso. Además, se está considerando la integración de algoritmos de aprendizaje profundo de más recientes, lo que promete mejorar aún más la precisión de los resultados obtenidos.

1.3 DESCRIPCIÓN DE LA TECNOLOGÍAS

El proceso de obtención y transformación de los datos se llevó a cabo utilizando Python, junto con las bibliotecas estándar para el tratamiento de datos, como Numpy y Pandas. En cuanto a la comparación de las herramientas de transcripción, se emplearon librerías como asr_evaluation (https://github.com/belambert/asr-evaluation), que evalúa la precisión de las transcripciones, además de NLTK (https://www.nltk.org/) y spaCy (https://spacy.io/).

Una vez seleccionada la herramienta más adecuada para la transcripción de las llamadas, se aplicaron diversas técnicas de machine learning según los casos de uso identificados. Al igual que en el análisis de imágenes, las técnicas de deep learning resultaron ser extremadamente útiles para trabajar con grandes volúmenes de texto. Para ello, se utilizaron las siguientes herramientas:

- A nivel de software, se trabajó principalmente con Python y PySpark, haciendo uso de bibliotecas de deep learning como Keras (https://keras.io/) y TensorFlow (https://www.tensorflow.org/).

- A nivel de hardware, se aprovecharon servidores locales para el procesamiento, junto con GPUs en la nube y en servidores on-premise, si bien algunos modelos entrenados no requieren de esta tecnología avanzada para su ejecución.

El proyecto se basó en tres tipos de modelos de machine learning. Inicialmente, se creó un modelo utilizando fastText (https://fasttext.cc/), desarrollado por Meta. En la siguiente fase, se probaron diferentes arquitecturas como redes convolucionales (CNN), redes LSTM (Long Short-Term Memory) bidireccionales y GRU (Gated Recurrent Units), todas ellas

implementadas con Keras. Finalmente, se llevó a cabo la experimentación con BERT (Bidirectional Encoder Representations from Transformers) y otros *transformers* derivados de BERT.

1.4 RESTRICCIÓN DE INFORMACIÓN

Este trabajo se ha realizado a partir de un proyecto real de Telefónica España. Por ese motivo, muchos de los datos incluidos en el mismo se muestran de forma cualitativa, pues se trata de información sensible. Por otro lado, no se incluyen nombres comerciales de los proveedores y colaboradores que han participado en el proyecto.

Capítulo 2. OBTENCIÓN DEL DATO

El primer paso en el desarrollo de modelos de Inteligencia Artificial es la obtención de la información de partida, por lo que esta tarea consume una parte destacable del total invertido en el proyecto.

El punto de partida son los audios con las llamadas. Éste es un dato interno de la compañía, pero su obtención no resulta tan fácil como podría pensarse. El proveedor responsable de la grabación y almacenamiento de las llamadas no dispone de opción de descarga masiva, aunque sí permite la descarga puntual, llamada a llamada. Gracias a esto, es posible crear un script que, mediante técnicas de web scraping, se ejecute sobre la web del proveedor y descargue un volumen suficiente de llamadas, una a una.

Como se aprecia en la Figura 2, la plataforma no sólo permite descargar el audio, también incluye una funcionalidad para transcribir y almacenar dicha llamada, por lo que es una de las herramientas incluida en la comparativa entre transcriptores. A lo largo del documento se la designa como servicio de transcripción 1.

Figura 2. Herramienta del proveedor que se encarga de grabación y almacenamiento de llamadas.

Se descargan las llamadas que cumplen, como mínimo, alguna de las siguientes características:

- Que se conozca la evaluación asignada por el usuario tras el contacto con el 1004.
- Que esté registrado si la llamada generó una venta para la empresa, particularmente en el caso de que el cliente contratase un producto de más valor respecto al que tenía previamente contratado.
- Con el objetivo de mejorar la calidad del servicio, la empresa monitoriza ciertas llamadas de los usuarios con el 1004. El equipo encargado de dicha tarea busca captar aquellos factores que más condicionan el éxito comercial.

Todo ello permite el desarrollo de modelos supervisados, pues se dispone de información adicional relativa al contacto telefónico, además de la propia transcripción.

Capítulo 3. HERRAMIENTAS DE TRANSCRIPCIÓN

3.1 INTRODUCCIÓN

Como se ha mencionado previamente, la fase inicial del proyecto incluye la comparativa de 4 herramientas de transcripción. Para poder realizar dicha comparativa, se transcriben 67 conversaciones manualmente, de modo que estas transcripciones actúen como patrón para medir el resultado de los softwares de la comparativa. Todas ellas son llamadas con una duración mínima, pero por lo demás la muestra está aleatorizada.

3.2 SISTEMAS EMPLEADOS

Como se ha mencionado, la obtención del dato (y su calidad) resultan clave en cualquier proyecto de esta envergadura. Por ello, el primer paso fue comparar cuatro herramientas de transcripción de audio a texto, con el objetivo de obtener la mayor calidad de dato posible. Las características de cada una de las cuatro herramientas son:

Servicio de transcripción 1: sistema actualmente utilizado en Telefónica España para grabar y transcribir ciertas llamadas. Es un proveedor con experiencia en la compañía.

Servicio de transcripción 2: Proyecto financiado por la Unión Europea, colaborado por Telefónica I+D y Telefónica Móviles.. El objetivo es analizar conversaciones en call centers europeos.

Servicio de transcripción 3: Software adaptado al lenguaje humano y el razonamiento, lo que permite crear tecnologías capaces de captar matices en las interacciones.

Servicio de transcripción 4: Tecnología Puntera mundial. Capaz de entender el lenguaje humano captando matices, responder preguntas complejas y mejorar con el uso.

El primer problema encontrado a la hora de realizar la comparativa entre las distintas herramientas fue el formato del output. Como se observa en Figura 3, el servicio 1 puede captar quién dice cada frase (agente comercial o cliente), así como asignar ciertas etiquetas para marcar la conversación. El formato obtenido es un HTML que se transforma en un fichero de texto plano con el contenido de la llamada.

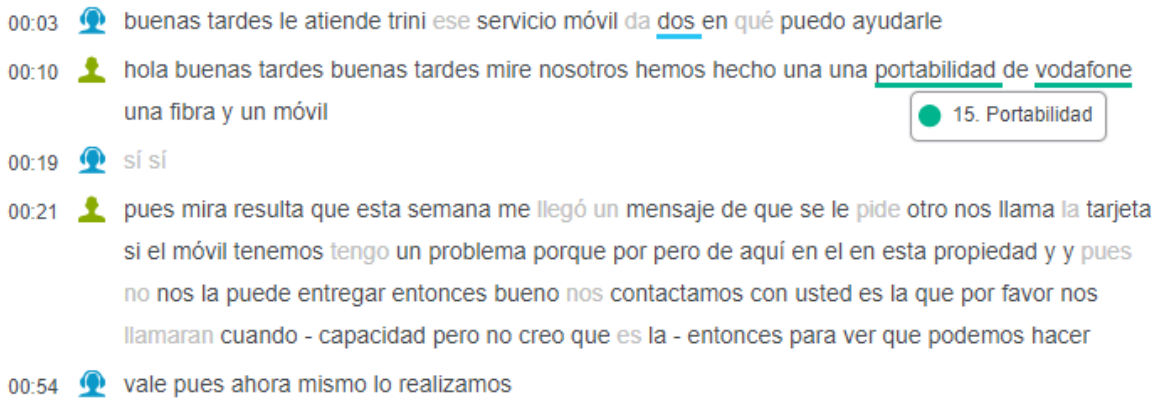

Figura 3. Interfaz web con la transcripción realizada por el servicio 1.

El servicio 2 no es capaz de distinguir interlocutores. La salida más avanzada que se obtiene es una relación entre las palabras transcritas, el momento en el que se comienza a decir esa palabra y el tiempo durante el cual se dice (Figura 4).

```
708300005805912_mono 3.07 0.31 buenas 1.00
708300005805912_mono 3.38 0.15 tardes 1.00
708300005805912_mono 3.53 0.07 le 1.00
708300005805912_mono 3.60 0.24 atiende 1.00
708300005805912_mono 3.84 0.35 vanesa 0.74
708300005805912_mono 4.19 0.10 de 0.98
708300005805912_mono 4.29 0.44 movistar 1.00
708300005805912_mono 4.73 0.11 en 1.00
708300005805912_mono 4.84 0.08 qué 1.00
708300005805912_mono 4.92 0.09 le 1.00
708300005805912_mono 5.01 0.26 podemos 1.00
708300005805912_mono 5.27 0.36 ayudar 1.00
708300005805912_mono 6.34 0.25 hola 1.00
708300005805912_mono 6.59 0.27 buenas 1.00
```

Figura 4. Output proporcionado por la herramienta 2

En relación con la tercera herramienta, el servicio dispone de un script que permite ejecutar el servicio en background en un servidor transcribiendo en serie un conjunto de audios. El formato de salida es similar al de la herramienta 2, asignando cada palabra al instante de tiempo en el que se dijo (Figura 5).

3.1	3.36	buenas
3.36	3.46	tarde
3.5	3.6	la
3.6	3.82	atiende
3.82	4.18	Vanesa
4.18	4.28	de
4.28	4.72	Movistar
4.72	4.82	en
4.82	4.92	qué
4.92	5.02	le
5.02	5.26	podemos
5.26	5.64	ayudar
6.3	6.6	hola
6.6	6.84	buenas
6.84	7.26	tardes
7.78	8	mira
8	8.02	llamaba

Figura 5. Output proporcionado por la herramienta 3.

Respecto al servicio de transcripción 4, ésta fue prácticamente transparente para nuestro equipo de trabajo. El proveedor recibió los audios correspondientes a las 67 conversaciones escogidas para realizar la comparativa entre sistemas de transcripción. Como se ve en la Figura 6, la plataforma es capaz de identificar temáticas dentro de la conversación, asignando una probabilidad a cada categoría.

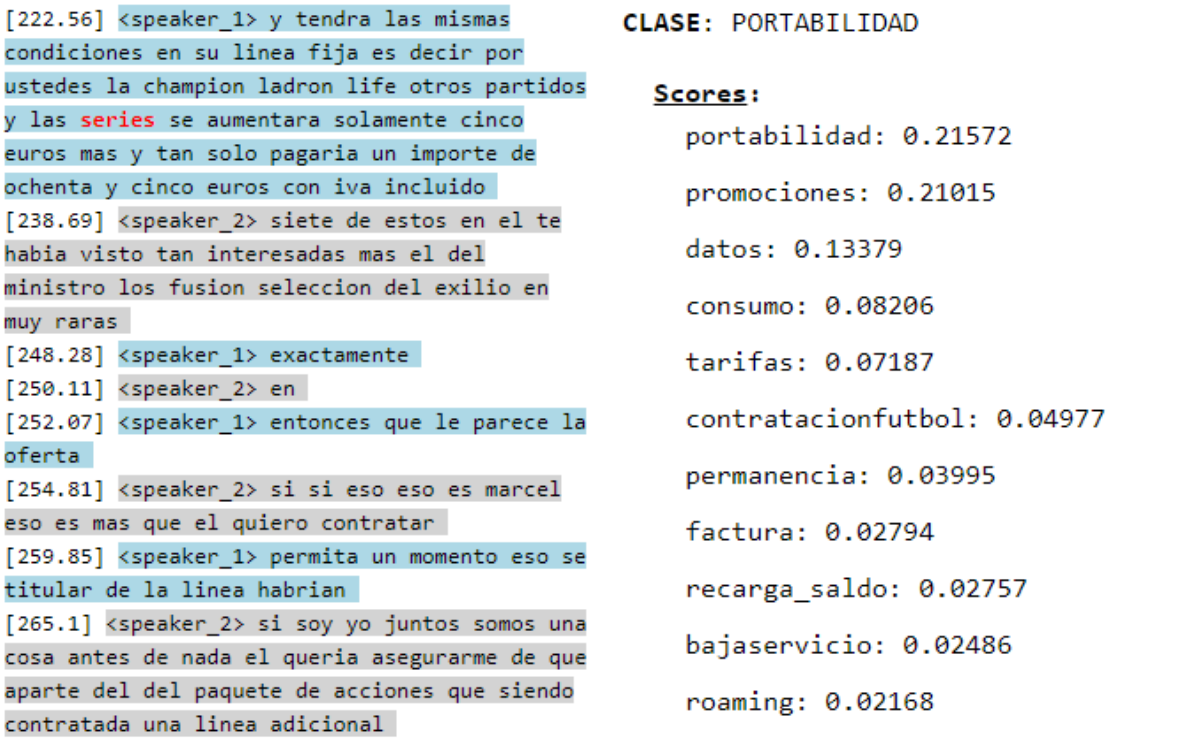

Figura 6. Interfaz web con la salida del servicio de transcripción 4, incluyendo el marcado de temáticas.

3.3 *FORMATO DE LA COMPARATIVA*

Para la comparativa, se trata cada conversación completa como una única cadena de texto. Esta aproximación busca simplificar la tarea, de modo que sea más sencillo comparar las distintas herramientas de transcripción, pues algunas de ellas son capaces de separar la conversación en oraciones y otras no.

Una de las principales fortalezas del sistema de transcripción número 3, en comparación con los sistemas 1 y 2, radica en su capacidad de adaptación específica al contexto del problema. Esto se logra gracias a dos características clave:

- **Ajuste por dominio**: Esta funcionalidad permite entrenar al sistema con un conjunto específico de transcripciones previamente seleccionadas, lo que afina su precisión en un entorno concreto. En este caso, se utilizaron tres colecciones diferentes: una con mil

grabaciones aleatorias del canal 1004 (se denomina ***Dom1k*** a lo largo del documento), otra con diez mil grabaciones también del 1004 (se denomina ***Dom10k*** a lo largo del documento), y una tercera compuesta por interacciones del canal 1002, orientado a soporte técnico, y recopilada por un equipo diferente (se denomina ***DomEY*** a lo largo del documento).

● **Palabras destacadas (add words)**: Este recurso permite indicar al sistema un conjunto de términos que deben recibir un tratamiento prioritario durante la transcripción. Es especialmente útil para nombres propios, marcas o vocabulario poco común. En este proyecto, por ejemplo, se incorporaron palabras como "Apple TV+", "miMovistar", "O2" y "Netflix", todas relevantes en el contexto tratado.

Durante el desarrollo, se probaron distintas configuraciones: sin personalización, con solo el ajuste por dominio, y combinando dominio y palabras clave. Dado que hay tres variantes de corpus disponibles, y cada una puede usarse con o sin palabras destacadas, el sistema 3 puede generar hasta seis versiones diferentes de una misma transcripción, además de una versión estándar sin ajustes: ***Dom1k***, ***Dom10k***, ***DomEY***, ***Dom1kAW***, ***Dom10kAW*** y ***DomEYAW***

Por su parte, los servicios 1 y 2 generan únicamente una transcripción por entrada, ya que carecen de mecanismos de personalización. El sistema 4 sí contempla una adaptación al dominio, aunque esta es gestionada automáticamente por el proveedor, sin intervención del equipo de trabajo.

3.4 TRATAMIENTO DEL TEXTO

Con el objetivo de reducir la complejidad en el procesamiento, se trata el texto obtenido para eliminar acentos y emplear caracteres alfabéticos en lugar de numéricos. Así, la frase "*contratar 2 líneas móviles adicionales*" se tratará como "*contratar dos lineas moviles adicionales*". Además, se eliminan también signos de puntuación.

3.5 MÉTRICAS

Con el objetivo de reducir la complejidad en el procesamiento, se trata el texto obtenido para eliminar acentos y emplear caracteres alfabéticos en lugar de numéricos. Así, la frase "*contratar 2 líneas móviles adicionales*" se tratará como "*contratar dos lineas moviles adicionales*". Además, se eliminan también signos de puntuación.

Para llevar una comparativa entre sistemas de transcripción es necesario establecer una métrica objetiva que evalúe la calidad de la salida de cada sistema. En este caso se han empleado dos: el Word Error Rate (WER) y la medida de similitud entre palabras mediante la librería SpaCy.

3.5.1 WORD ERROR RATE (WER)

El WER (https://es.wikipedia.org/wiki/Word_Error_Rate) es una medida utilizada frecuentemente en la evaluación de sistemas de transcripción y traducción automática. Compara una transcripción de referencia (en nuestro caso, la manual) con la que se desea evaluar, de modo que determina las palabras que se sustituyen por otras (S), las que se eliminan (B) y las palabras que se incluyen de forma errónea (I); y las relaciona con el número total de palabras en el fragmento evaluado (N). La expresión matemática se muestra en la Ecuación 1.

$$WER = \frac{S + B + I}{N}$$

Ecuación 1. Definición del Word Error Rate (WER)

El cálculo del WER se realiza con la librería asr_evaluation, que también calcula la relación de palabras correctamente transcritas y el total de palabras en el texto (Word Recognition Rate, WRR), y la relación entre oraciones incorrectas y oraciones totales (Sentence Error Rate, SER). La expresión matemática de estas dos métricas se muestra en la Ecuación 2 y Ecuación 3, respectivamente.

$$WRR = \frac{Número\ de\ palabras\ correctamente\ transcritas}{Número\ de\ palabras\ totales\ en\ la\ conversación}$$

Ecuación 2. Definición del Word Recognition Rate (WRR)

$$SER = \frac{Número\ de\ oraciones\ correctamente\ transcritas}{Número\ de\ oraciones\ totales\ en\ la\ conversación}$$

Ecuación 3. Definición del Sentence Error Rate (SER)

Se decide emplear únicamente el WER por ser la métrica que más se usa habitualmente. El SER no se puede emplear pues, como se ha explicado, no se mantiene la distinción entre frases y las conversaciones son tratadas como palabras sucesivas, sin signos de puntuación.

La librería devuelve el texto de referencia indicando en rojo y mayúsculas aquellas palabras que son erróneas en la transcripción, como se aprecia en la Figura 7. También calcula las métricas descritas previamente (Figura 8).

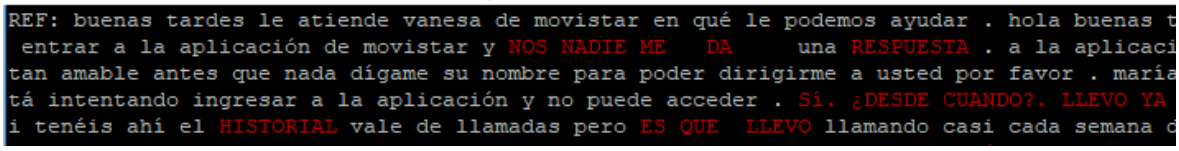

Figura 7. Ejemplo de salida proporcionado por la librería asr_evaluation

```
SENTENCE 1
Correct          =  71.6%   867   (  1211)
Errors           =  29.2%   354   (  1211)
Sentence count: 1
WER:    29.232% (       354 /       1211)
WRR:    71.594% (       867 /       1211)
SER:   100.000% (         1 /          1)
```

Figura 8. Métricas que devuelve la librería asr_evaluation, en la comparativa entre dos transcripciones

Mediante un script se extiende al conjunto de llamadas de la comparativa el cálculo del WER. Como se ha visto en la Ecuación 1, una transcripción perfecta tendrá un WER de 0.

$$Word\ Accuracy = 1 - \frac{S + B + I}{N} = 1 - WER$$

Ecuación 4. Definición del Word Accuracy

A partir del WER se calcula el Word Accuracy (Ecuación 4). Esta métrica facilita la interpretación de los resultados: un Word Accuracy de 1 implica que la transcripción del sistema coincide exactamente con la transcripción manual. **Aunque en los capítulos sucesivos se hace referencia habitualmente al WER, se refiere realmente al Word Accuracy en tanto por 1.**

En la Tabla 1 se incluye un conjunto de resultados parciales para ciertos servicios:

identificador	Serv_1	Serv_2	Serv_3	Serv_3_Dom1k
XXXXXXXXXX	0.51053	0.36143	0.44084	0.45138
XXXXXXXXXX	0.56336	0.44521	0.44949	0.47432
XXXXXXXXXX	0.45312	0.10938	0.17188	0.35938
XXXXXXXXXX	0.66310	0.65107	0.66444	0.67380

Tabla 1. Muestra de resultados en el cálculo del WER para una pequeña muestra de llamadas y servicios de transcripción.

3.5.2 MEDIDA DE SIMILITUD CON SPACY

SpaCy es una biblioteca de NLP escrita en Python y Cython. Rivaliza directamente con soluciones más tradicionales, como la librería NLTK, aunque está más enfocada a situaciones en producción. Permite extraer entidades y etiquetas de textos, así como desarrollar procesos de natural language understanding (NLU). También resulta muy útil en el preprocesado del texto previo a la creación de modelos de machine learning, al permitir la lematización, la tokenización o la eliminación de las stopwords.

En este caso se ha hecho uso de otra funcionalidad de la librería, la similitud (https://spacy.io/usage/vectors-similarity), que permite evaluar la distancia existente entre

palabras o textos. El objetivo consiste, lógicamente, en ver qué sistema de transcripción devuelve un resultado con una similitud mayor respecto a la transcripción manual.

La medida se basa en la comparación entre la representación vectorial entre dos textos. Se parte de un word embedding: representación de una palabra a partir de un vector de dimensión determinada (https://es.wikipedia.org/wiki/Word_embedding). En la Figura 9 se muestra un ejemplo sencillo con la representación vectorial, en tres ejes, de las palabras "coche", "perro" y "gato". En la realidad la dimensionalidad de los vectores es mucho mayor, pudiendo llegar a superar la dimensión 500. Al tratar cada palabra como un vector, es posible medir matemáticamente la distancia entre palabras.

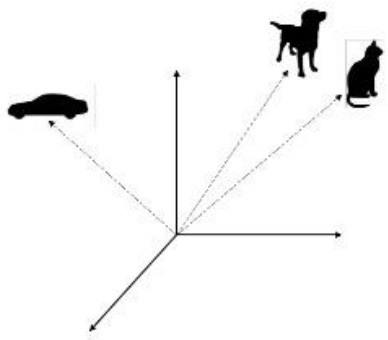

Figura 9. Ejemplo del concepto de word embedding.

(Fuente: https://www.slideshare.net/BhaskarMitra3/a-simple-introduction-to-word-embeddings)

Aunque la herramienta permite entrenar embeddings de acuerdo con el corpus empleado, en este caso se hace uso de los vectores preentrenados incluidos en la propia librería. SpaCy extrae el vector asociado a cada una de las palabras de una conversación, y asocia la media de estos vectores a dicha conversación. De este modo, obtiene una representación vectorial del texto completo, lo que posibilita medir la distancia entre la transcripción realizada manualmente y la realizada con alguno de los servicios de transcripción.

En la Tabla 2 se aprecia un conjunto de llamadas a modo de ejemplo. Nótese que un valor 1 representa semejanza total entre textos.

identificador	Serv_1	Serv_2	Serv_3	Serv_3_Dom1k	Serv_3_Dom10k
XXXXXXXXXX	0.9340	0.9876	0.9892	0.9795	0.9795
XXXXXXXXXX	0.9979	0.9954	0.9959	0.9828	0.9828
XXXXXXXXXX	0.9940	0.9939	0.9941	0.9910	0.9910
XXXXXXXXXX	0.9937	0.9836	0.9796	0.9250	0.9250

Figura 10. Muestra de resultados en el cálculo de la similitud para una pequeña muestra de llamadas y servicios de transcripción

3.6 RESULTADOS DE LA COMPARATIVA

Una vez tratado el texto y definidas las métricas a emplear, se evalúa el rendimiento de todas las opciones de transcripción consideradas.

Como se aprecia en la Figura 11, los mejores resultados en cuanto al WER se consiguen al emplear un conjunto de 10000 llamadas (Dom10k), pues esta variante consigue los valores más altos en alrededor de 50 ocasiones. El uso de Add Words no parece influir demasiado en el resultado, pero se decide mantener ya que podría ayudar a identificar términos como marcas comerciales, lo que resulta de interés.

En cuanto a la similitud, por el contrario, es la versión estándar de la herramienta de transcripción 3 es la que consigue una similitud más cercana a la transcripción manual en la mayoría de casos. En esta métrica, no obstante, se observa menos variabilidad en los resultados que en el caso del WER.

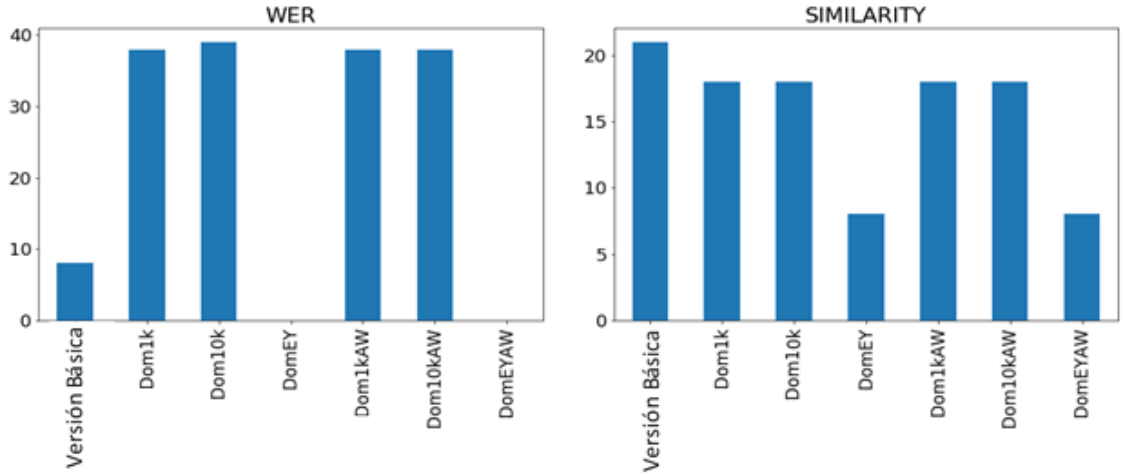

Figura 11. Métrica WER y similitud para las distintas variantes del servicio de transcripción 3, evaluando el conjunto de llamadas de la comparativa.

Para poder emplear un criterio de decisión objetivo entre el servicio de transcripción 3 en su versión básica (sin dominio ni add words) y Dom10kAW, se comparan los resultados obtenidos solamente teniendo en cuenta estas dos variantes.

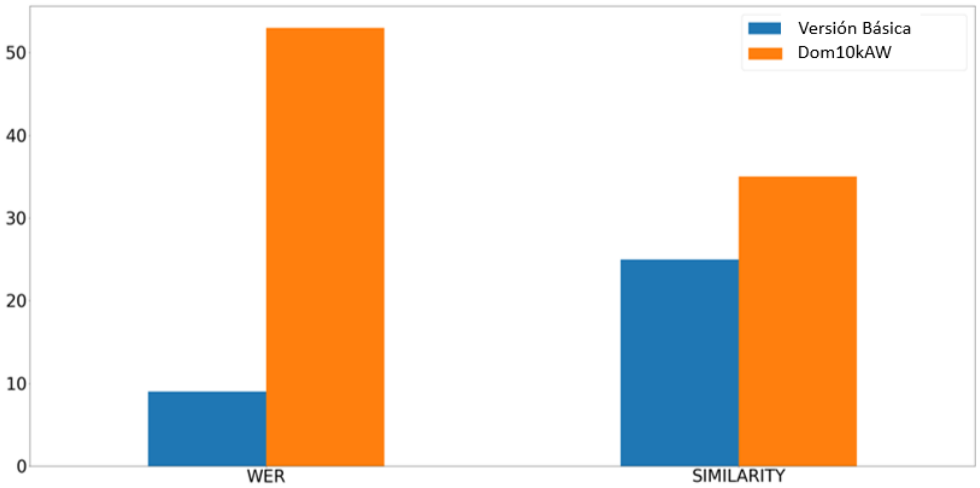

Figura 12. Métricas WER y similitud comparando la versión básica del servicio 3 y el modelo ajustado con 10.000 llamadas y add words (Dom10kAW).

En la Figura 12 se aprecia cómo el uso del dominio y las add words mejoran claramente los resultados. Esta tendencia se mantiene en el caso de la similitud, pero los resultados son más ajustados.

Definida la variante del servicio 3 a emplear, Dom10kAW, se puede comparar ésta con el resto de soluciones propuestas.

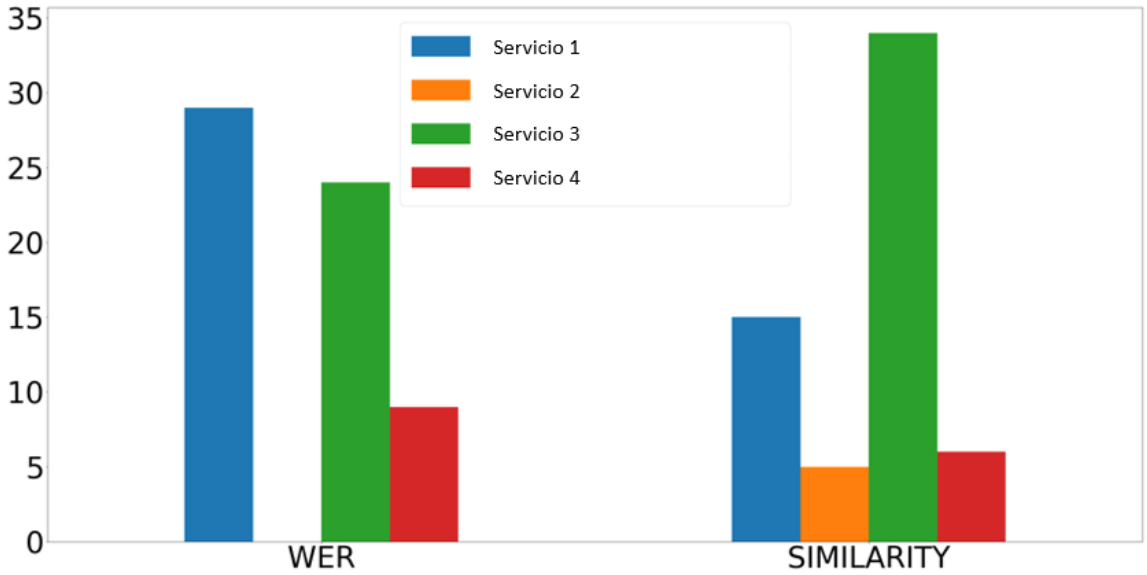

Figura 13. Resultados obtenidos con las cuatro herramientas de la comparativa.

En la Figura 13 se representa el número de transcripciones en las que lidera cada herramienta (si hay empate entre dos soluciones se suma una unidad a cada una). Se puede ver que el servicio 1 y 3 son los que consiguen mejores resultados, aunque el ganador depende de la métrica considerada. El proveedor 4, que partía con unas expectativas muy prometedoras, se queda ligeramente por detrás. La herramienta 2, por su lado, consigue resultados considerablemente peores.

Los diagramas de cajas y bigotes de la Figura 14 muestran la distribución de datos del WER, para las cuatro herramientas. En los resultados destaca la variabilidad en los valores del servicio 4.

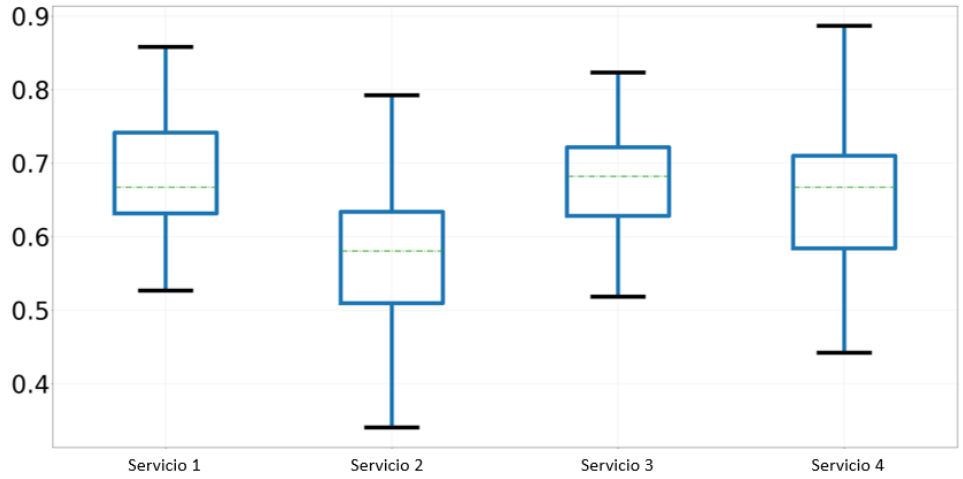

Figura 14. Diagrama de cajas y bigotes con datos de WER, para cada servicio.

De modo similar, en la Figura 15 se muestran los resultados de similitud. Como era esperable, existe una mayor proximidad entre las distintas soluciones. Destaca, de nuevo, la variabilidad en los resultados del servicio 4, que consigue prácticamente los mejores valores en ciertas conversaciones, mientras que en otras alcanza los valores más bajos. Algo similar ocurre con el primer servicio, aunque en este caso la diferencia entre el valor mínimo y máximo es menor.

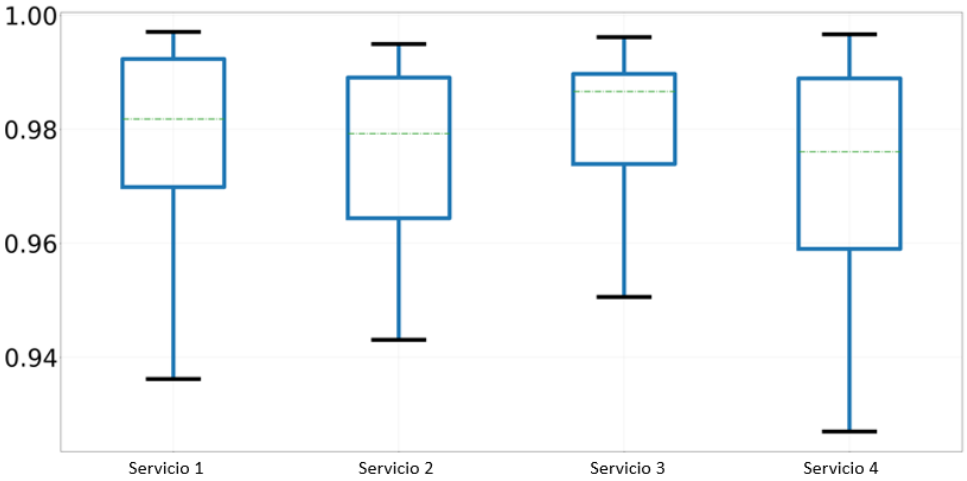

Figura 15. Diagrama de cajas y bigotes con datos de similitud, para cada servicio.

Sobre la comparativa entre diferentes servicios se aplican test estadísticos (Tabla 2 para la métrica WER y Tabla 3 para la métrica similitud).

	diff	lwr	upr	p adj
Serv4-Serv2	0.081919000	0.040022960	0.12381504	0.0000050
Serv3_Dom10kAW-Serv2	0.108180500	0.066284460	0.15007654	0.0000000
Serv1-Serv2	0.114480333	0.072584293	0.15637637	0.0000000
Serv3_Dom10kAW_Serv4	0.026261500	-0.015634540	0.06815754	0.3682903
Serv1-Serv4	0.032561333	-0.009334707	0.07445737	0.1868464
Serv1-Serv3_Dom10kAW	0.006299833	-0.035596207	0.04819587	0.9799576

Tabla 2. Test estadístico aplicado a la comparativa de servicios de transcripción para la métrica WER.

	diff	lwr	upr	p adj
Serv4-Serv2	0.000070000	-0.019918470	0.02005847	0.9999997
Serv3_Dom10kAW-Serv2	0.013473333	-0.006515137	0.03346180	0.3033132
Serv1-Serv2	0.010576667	-0.009411804	0.03056514	0.5200019
Serv3_Dom10kAW-Serv4	0.013403333	-0.006585137	0.03339180	0.3078963
Serv1-Serv4	0.010506667	-0.009481804	0.03049514	0.5257218
Serv1-Serv3_Dom10kAW	-0.002896667	-0.022885137	0.01709180	0.9819867

Tabla 3. Test estadístico aplicado a la comparativa de servicios de transcripción para la métrica similitud.

Como se aprecia, las conclusiones dependerían, además, de la métrica que se emplease. En el caso del WER las diferencias entre el servicio 2 y el resto son estadísticamente significativas, mientras que el test no permite alcanzar conclusiones en la comparativa entre el resto de servicios.

Respecto a la métrica a emplear, existen ciertas razones para dar más peso al WER frente a la similitud. La similitud se basa en la medida de la distancia entre los dos vectores que representan cada uno a una transcripción, la manual y la que se quiere evaluar. Cada vector resulta de calcular la media de vectores que representan las palabras de la transcripción (típicamente hay unas 1000 palabras por cada transcripción). Esto implica que, al calcular la media con tantos elementos, los vectores que representan a dos transcripciones suelen ser similares, aun cuando la temática del texto cambia. De hecho, se realiza la prueba de medir la similitud entre transcripciones manuales de conversaciones totalmente distintas, y en varios casos se obtuvo una similitud superior al 90%.

Como se ha explicado, la selección del servicio para transcribir masivamente las llamadas excede las competencias del equipo de trabajo.

Capítulo 4. CASOS DE USO Y MODELOS

4.1 POSIBLES CASOS DE USO

Como se ha explicado, el objetivo del proyecto consiste en analizar las interacciones entre Telefónica y los consumidores a través del canal 1004. Este análisis puede dar lugar a diferentes casos de uso, en este proyecto se definen dos:

Detección de insatisfechos: El objetivo es ordenar a las personas que llamaron al 1004 el día anterior en función del grado de insatisfacción. Esto permite optimizar las acciones de la empresa para mejorar la satisfacción de los clientes. El proyecto se centra en este caso de uso.

Detección de llamadas con propensión a venta: El objetivo es entrenar un modelo partiendo de las llamadas de las cuales se sabe si han terminado en venta o no. De este modo, se puede detectar las llamadas del día previo que, aunque no han producido una venta hasta ese momento, tienen una alta probabilidad de hacerlo así que se debería contactar de nuevo al posible cliente lo antes posible. El objetivo sería generar un listado de potenciales clientes, ordenados en base a su probabilidad de alta.

4.2 DETECCIÓN DE INSATISFECHOS

Como parte del proceso de atención al cliente, tras una interacción con el canal 1004, los usuarios suelen recibir días después una llamada automatizada invitándolos a calificar el servicio recibido. Esta práctica permite a Telefónica detectar rápidamente posibles casos de insatisfacción y actuar en consecuencia. Sin embargo, tal como se ilustra en la Figura 16, esta metodología enfrenta una limitación importante: la tasa de participación es baja, lo que deja sin información directa sobre el nivel de satisfacción de una parte significativa de los usuarios.

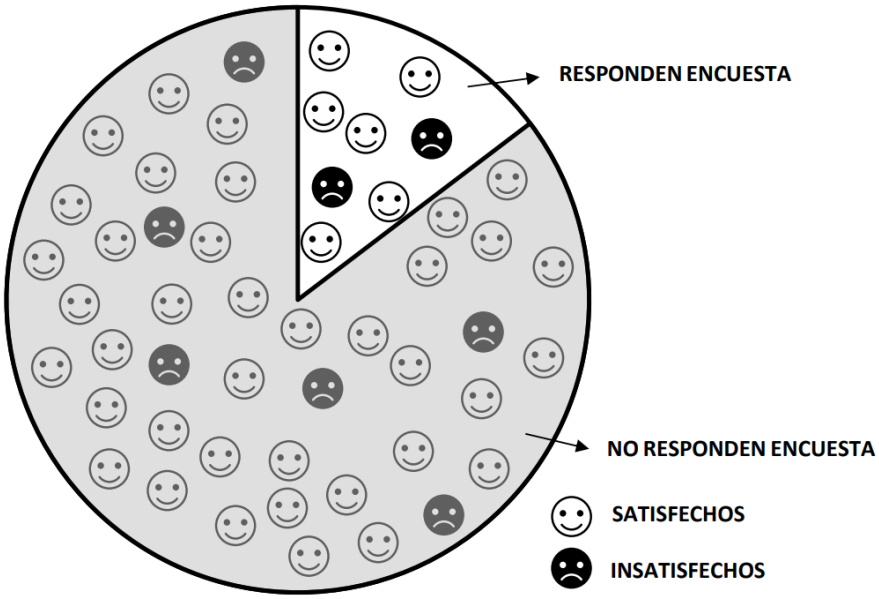

Figura 16. La población de estudio son los clientes que llaman al 1004. Sólo una parte de éstos responden a la encuesta posterior

Pese a la escasa proporción de respuestas, los datos analizados por Telefónica indican que la muestra es suficientemente representativa como para mantener la proporción entre opiniones positivas y negativas. En este contexto, el valor añadido del modelo desarrollado radica en su capacidad para identificar clientes descontentos únicamente a partir de la conversación mantenida con el agente, sin depender de que estos respondan posteriormente a la encuesta.

El propósito es claro: detectar de manera anticipada señales de insatisfacción y tomar medidas correctivas que contribuyan a mejorar la experiencia del cliente, ayudando así a reducir la tasa de abandono.

Para entrenar este sistema, se utiliza un conjunto de grabaciones de llamadas en las que sí se dispone de la valoración aportada por el usuario. Estas valoraciones, en una escala de 0 (muy insatisfecho) a 10 (muy satisfecho), permiten desarrollar un modelo supervisado. El enfoque aplicado se basa en clasificación binaria: se considera que un usuario está satisfecho si la puntuación supera el 5, e insatisfecho si es igual o inferior.

4.2.1 CONJUNTO DE DATOS

En este punto del proyecto, y tal como se ha explicado, aún no se ha seleccionado una herramienta de transcripción a emplear, únicamente se realiza la comparativa para facilitar esta decisión. Es por ello que se dice emplear el servicio 2.

El servicio 2 logra unos resultados modestos, pero es una herramienta interna lo que facilita acceder a todas las funcionalidades. Además, permite el uso de Spark en el cluster on Premise de la compañía, por lo que permite obtener resultados de forma ágil.

4.2.2 EVALUACIÓN DE LOS MODELOS

Para evaluar el desempeño de un modelo se parte de la información incluida en la matriz de confusión (Tabla 4). La primera métrica que se emplea es la accuracy (Ecuación 5), esto es, la relación entre los aciertos de un modelo al clasificar un conjunto de datos y el tamaño del propio conjunto.

		PREDICCIÓN	
		CLIENTE SATISFECHO	CLIENTE INSATISFECHO
REALIDAD	CLIENTE SATISFECHO	Verdaderos Negativos (TN)	Falsos Positivos (FP)
	CLIENTE INSATISFECHO	Falsos Negativos (FN)	Verdaderos Positivos (TP)

Tabla 4. Matriz de confusión

$$Accuracy = \frac{TN + TP}{TN + FP + FN + TP}$$

Ecuación 5. Definición de accuracy

Si bien en el proyecto no es posible concretar beneficios económicos y costes, pues es información corporativa restringida, sí se puede saber que el beneficio obtenido por la gestión de un cliente insatisfecho es mayor al coste de gestión de un cliente (Ecuación 6).

$$\frac{Beneficio\ gestion\ insatisfecho}{Coste\ gestión\ cliente} > 1$$

Ecuación 6. Relación beneficio gestión insatisfecho y coste de gestión

Esto implica que en un modelo no sólo se debe maximizar la accuracy, sino que se deben tener en cuenta otras métricas asociadas a falsos positivos (FP) y falsos negativos (FN): precisión y exhaustividad (denominadas precision y recall en inglés, respectivamente).

Precision: Relaciona el número de clientes que el modelo acierta al clasificar como insatisfechos con la cantidad total de insatisfechos que predice el modelo, acierte o no (Ecuación 7).

$$Precision = \frac{TP}{TP + FP}$$

Ecuación 7. Definición de precisión

Como se aprecia, un valor elevado de precisión se asocia con un número de falsos positivos bajos. Esto implica que, cuando se predice que un cliente está insatisfecho, se tiende a acertar.

Recall: Relaciona el número de usuarios que el modelo acierta al clasificar como insatisfechos con la cantidad total real de insatisfechos (Ecuación 8).

$$Recall = \frac{TP}{TP + FN}$$

Ecuación 8. Definición de exhaustividad (recall)

Como se aprecia, un valor elevado de recall se asocia con un número de falsos negativos bajos. Esto implicaría que, cuando se predice que un consumidor está satisfecho, se tiende a acertar. Es decir, rara vez se clasifica como satisfecho a un usuario que realmente no lo está.

Empleando la precisión y el recall, el beneficio obtenido se puede representar matemáticamente como:

$$Totales * Recall\ [Ganancia - Coste\ llamada\ \left(\frac{1 - Precision}{Precision}\right)]$$

Ecuación 9. Cálculo del beneficio obtenido por la gestión de insatisfechos.

En la Ecuación 9, "Totales" se refiere al número total de usuarios que no han respondido a la encuesta y de los cuales no se conoce la satisfacción, y "Ganancia" a la diferencia entre el impacto que tiene la gestión de un individuo insatisfecho y el coste de contactar con dicho individuo ("Coste llamada").

En la Ecuación 9 se puede obtener el valor mínimo de precisión para obtener beneficios si se impone que el resultado de la Ecuación 9 sea positivo o nulo (Ecuación 10).

$$Precision \geq \frac{Coste\ llamada}{Ganancia + Coste\ llamada}$$

Ecuación 10. Precisión mínima para obtener beneficio

Matemáticamente, la precisión es el parámetro principal a maximizar para obtener beneficio. De manera intuitiva se puede ver que en el caso de uso que se contempla, también podría interesar tener un recall elevado a costa de perder precisión, pues esto implica que en pocas ocasiones se clasificaría un cliente como satisfecho de manera errónea. Puesto que el beneficio obtenido por tratar a un usuario no satisfecho supera con creces el coste de contacto, es preferible que el modelo tienda a sobreestimar el número de insatisfechos. Esta idea se refuerza al representar los beneficios potenciales en función de la precisión y el recall, en la Figura 17.

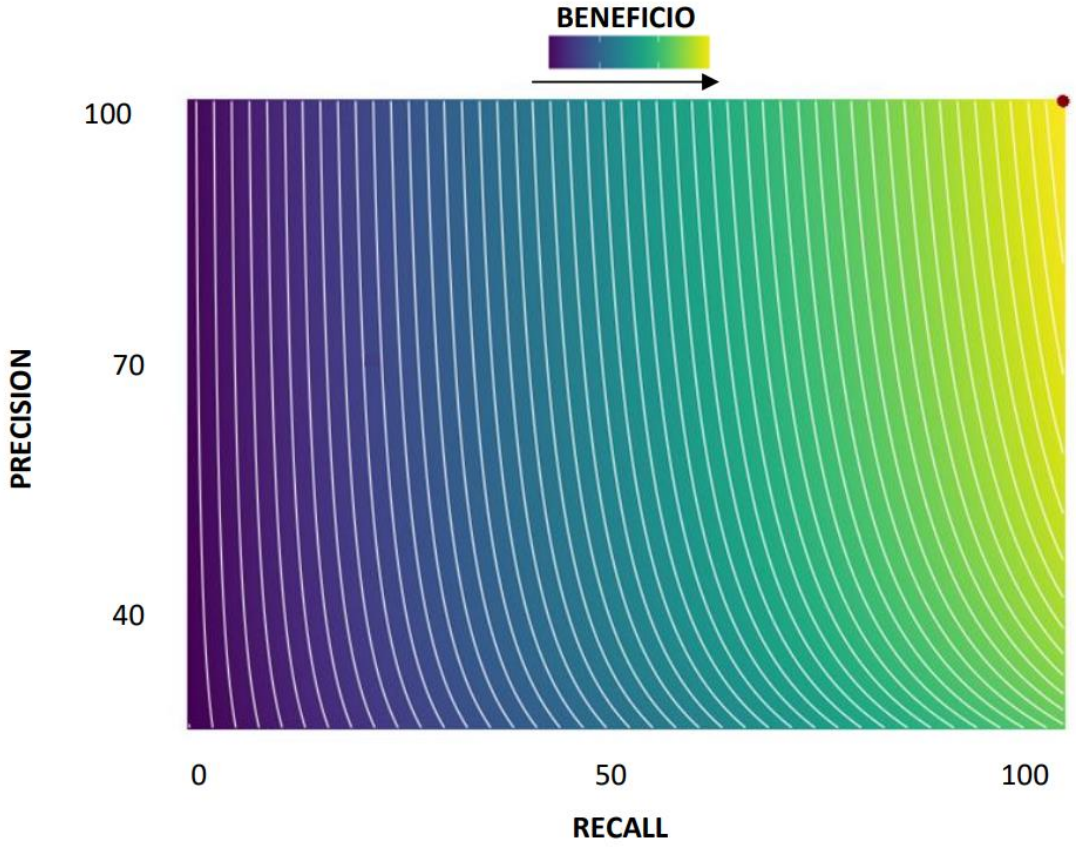

Figura 17. Variación del beneficio con precisión y recall.

Desde un punto de vista económico, también se debe tener en cuenta que un mayor valor de recall puede incrementar considerablemente los costes, haciendo disminuir el retorno de la inversión (ROI) respecto a modelos con menor recall y más precisión. En la valoración económica del modelo empleado, que queda fuera del alcance del proyecto, se deberían tener especialmente en cuenta estos factores.

Por último, los modelos de clasificación suelen devolver una probabilidad de que un cierto registro pertenezca a una clase u otra. El umbral de clasificación es el valor límite de dicha probabilidad a partir del cual el modelo clasifica un registro como de una clase u otra, por lo que es un parámetro muy relevante, pues condiciona las métricas explicadas. En este proyecto es establece en un valor típico: 0.5.

4.3 INTRODUCCIÓN AL TRATAMIENTO DE TEXTO EN MACHINE LEARNING

4.3.1 BAG OF WORDS (BOW)

En machine learning, los modelos basados en texto requieren la modelización de éste de una forma adecuada para la creación de algoritmos. Una de las técnicas tradicionales es el modelo Bag of Words (BoW).

En esta aproximación se construye un vocabulario compuesto por las palabras que aparece en el corpus empleado. Cada documento se representa por un vector que tiene en cuenta la cantidad de veces que una palabra aparece incluida en dicho documento. En la Figura 18 se representa el vector asociado a la oración "quiero una línea móvil", basada en un vocabulario muy simple a modo de ejemplo.

Vocabulario	factura	línea	una	fibra	móvil	datos	quiero
Quiero una línea móvil	0	1	1	0	1	0	1

Figura 18. Modelización de una oración según el modelo Bag of Words

En un caso real, un documento vendrá representado por un vector de dimensión igual al corpus empleado (típicamente cientos o miles de palabras). No obstante, la mayoría de los elementos del vector serán ceros (vectores "sparse", en inglés). Todo ello genera un coste computacional elevado.

Por otro lado, el modelo BoW no tiene en cuenta el orden de las palabras en la oración ni mantiene ninguna relación semántica o de contexto entre las oraciones ni las palabras.

4.3.2 WORD EMBEDDING

Para resolver los problemas vistos en el apartado 4.3.1, se desarrollaron los word embeddings. Se trata de una representación de las palabras capaz de capturar la similitud entre éstas, de modo que se pasa de una simple representación numérica a la creación de un

espacio vectorial en el que la representación de dos palabras similares o del mismo ámbito ("fibra", "adsl") tienen una cercanía entre ellas mayor que otras de ámbitos dispares ("adsl", "avión"). Otra de las ventajas es que el uso de vectores permite agregar palabras. Por ejemplo, el resultado de "hombre" + "monarquía" debería devolver un vector cercano en el espacio a "rey".

La forma más habitual de obtener el conjunto de vectores se basa en el uso de redes neuronales poco profundas (*shallow neural networks* en inglés), que procesando un conjunto suficientemente grande de datos consigue extraer la representación vectorial de cada palabra incluida, manteniendo la relación entre éstas (Figura 9). Otra de las características de los word embedding es que son modelos no supervisados, pues no necesitan etiquetas asociadas a los distintos textos para generar la distribución de vectores.

Una de las herramientas más populares para generar la distribución de vectores es Word2Vec (2), de Google. Incluye dos métodos:

- Continuous Bag Of Words (CBOW): En esta aproximación se entrena una red para predecir una palabra a partir de las palabras del contexto.
- Skip-gram: En este caso el planteamiento es el inverso: la red se entrena para, a partir de una cierta palabra, predecir las palabras del contexto.

La principal diferencia entre ambos métodos (Figura 19) reside en que skip-gram toma como input las palabras individuales, mientras que CBOW procesa los contextos, por lo que una misma palabra puede aparecer en dos contextos distintos. Además, skip-gram parece haber dado mejores resultados a la hora de manejar palabras poco frecuentes. En la mayoría de los casos, ambos métodos dan resultados similares (3).

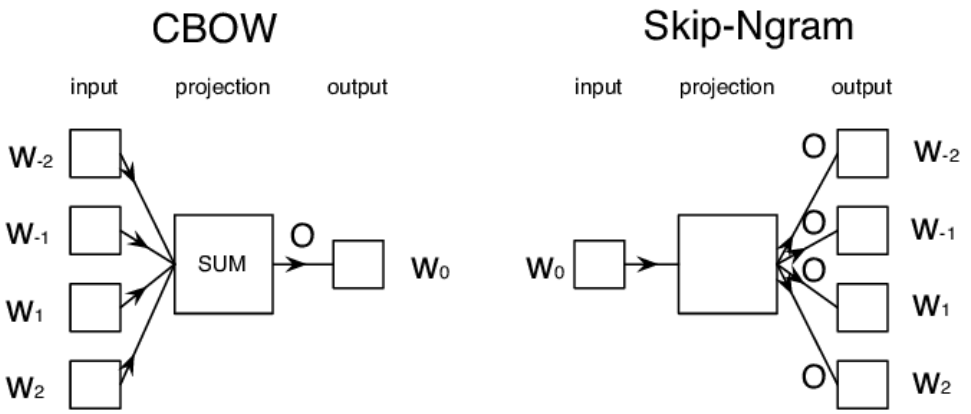

Figura 19. Esquema de CBOW y Skip-gram para generar el word embedding (Fuente: https://www.researchgate.net/figure/Illustration-of-the-Skip-gram-and-Continuous-Bag-of-Word-CBOW-models_fig1_281812760)

4.4 FASTTEXT

En 2016 Meta lanzó la librería open source fastText (4), una variante de Word2Vec. Está optimizada para CPUs, por lo que resulta muy útil si no se dispone de GPUs. Escrita en C++, destaca por ser un framework muy rápido, capaz de generar los embeddings de un vocabulario de un billón de palabras en 10 minutos (5).

Pese a que lleva un tiempo sin actualizarse (su última versión fue lanzada en 2020, incluyendo mejoras en el ajuste automático de hiperparámetros y en la API de Python), es una herramienta rápida y eficiente, óptima para etapas tempranas en proyectos de esta naturaleza.

La principal diferencia con Word2Vec es que fastText divide las palabras en n-gramas, esto es, partes de una palabra que en conjunto tienen sentido, pero no de forma individual. Este parámetro es configurable. Por ejemplo, si se fija un valor de 3, la palabra "silla" quedaría representada por los trigramas: "<si", "sil", "ill", "lla" y "la>". ">" y "<" se emplean para diferenciar un n-grama de una palabra cuando comparten la representación, como el caso de "la" y "la>", que tienen distinto significado. Para evitar un coste computacional demasiado elevado, fastText incluye funcionalidades para limitar el tamaño del conjunto de ngramas

empleado. Se debe tener en cuenta que la herramienta no valora el orden de los ngramas en una oración, sólo qué ngramas y palabras aparecen (6). El vector asociado sería, por tanto, la combinación de los vectores de los distintos ngramas. La principal ventaja del uso de n-gramas es que facilita la vectorización de palabras poco frecuentes, pues divide estas en conjuntos de ngramas cuya frecuencia en el corpus es mayor. Esto ayuda a conseguir mejores resultados que word2vec cuando se trata de representar una palabra desconocida, no incluida en el vocabulario (7).

El framework incluye una funcionalidad para clasificación de textos. Una oración viene representada por los vectores asociados a los ngramas de la misma y a la propia palabra. Esto permite asociar también un vector a cada etiqueta. Así, como se aprecia en la Figura 20, las palabras de una cierta temática tendrán representaciones similares a la propia etiqueta.

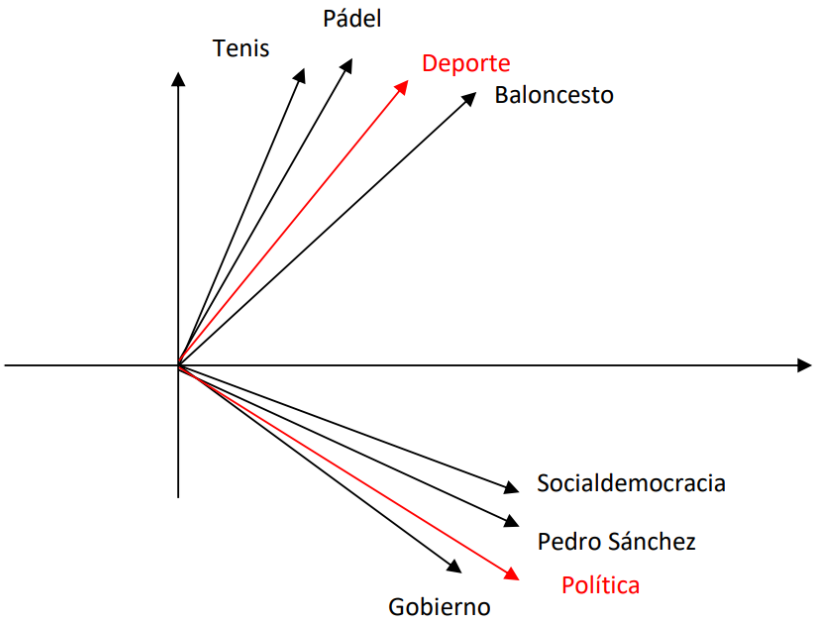

Figura 20. Representación vectorial de ciertas palabras y su etiqueta asociada.

Durante el entrenamiento, fastText ajusta los pesos que asigna a cada ngrama y a la palabra completa mediante un descenso estocástico del gradiente. El ajuste de estos pesos permite incrementar la relación entre la representación de las palabras de un texto y su etiqueta, de modo que el modelo aprende a clasificar correctamente las oraciones.

Tanto fastText como el resto de modelos que se verán en epígrafes sucesivos se caracterizan por tener un elevado número de parámetros, cuyo valor afecta de manera notable a los resultados obtenidos. Tal como se ha explicado, fastText incluye desde hace varios años funcionalidades para la optimización de hiperparámetros. Pese a ello, los responsables del proyecto optaron por incorporar una herramienta específica para esta finalidad. La elegida fue Optuna.

Optuna (https://optuna.org/) es una librería open source basada en algoritmos de optimización bayesiana que permite la búsqueda de hiperparámetros de modelos de machine learning. Se basa en ejecutar pruebas sucesivas variando los valores empleados, de modo que cada prueba aprende de las previas y tiende al resultado óptimo. Optuna destaca por permitir lanzar pruebas en paralelo cuyos resultados se escriben en una base de datos (en el caso del proyecto una base de datos PostgreSQL) compartida, de modo que cuando se lanza una nueva configuración de hiperparámetros se tienen en cuenta todas las anteriores. Es una herramienta muy popular en proyectos de machine learning, y está en constante evolución. Su última actualización se produjo en enero de 2025, con mejoras en la integración de nuevos algoritmos y en la distribución a gran escala.

Para el ajuste del modelo de fastText se decide emplear el 80% de los datos disponibles, mientras que el 20% restante se emplea en validación. Para incrementar la fiabilidad a la hora de fijar los hiperparámetros se emplea validación cruzada con tres folds. Como se ha explicado, el número de usuarios satisfechos supera con creces a los insatisfechos, por lo que se eliminan de forma aleatoria parte de los registros con satisfacción positiva.

Las pruebas de Optuna se lanzan en paralelo en el cluster de la compañía, empleando PySpark. Tras un conjunto de pruebas, el mejor modelo alcanzado obtiene los resultados de la Tabla 5.

Modelo	Accuracy entrenamiento	Accuracy test	Precision	Recall
fastText	0,8596	0,8392	0,5977	0,1201

Tabla 5. *Resultados del mejor modelo obtenido con fastText.*

En el proyecto se ha optado por emplear como primer modelo fastText por ser un framework rápido y, especialmente, por no requerir el uso de GPUs. En las fases iniciales del proyecto este hardware no estaba disponible, por lo que fastText facilitó tener un primer modelo para valorar de forma preliminar la viabilidad el caso de uso, así como conseguir un primer resultado para comparar los modelos sucesivos que sí requieren de GPUs.

4.5 DEEP LEARNING

El aprendizaje profundo (deep learning en inglés) es una variante del machine learning basado en el procesado de datos y extracción de patrones siguiendo una lógica similar a la empleada por el cerebro humano. En la práctica existen múltiples tipos de estructuras que permiten desarrollar procesos de aprendizaje profundo.

4.5.1 ¿POR QUÉ USAR DEEP LEARNING?

Aunque fastText obtiene unos resultados razonablemente buenos, se trata un modelo optimizado para el uso de CPUs. Los avances más prometedores dentro del NLP en los últimos tiempos requieren del uso de deep learning, lo que implica usar potentes GPUs.

Por otro lado, Telefónica España mantiene una apuesta clara por la innovación, por lo que la empresa tiene especial interés en evaluar las últimas novedades en el entorno Big Data, tanto en términos de software como de hardware.

Por esta razón, en el proyecto se hace uso de dos tipos de hardware:

- Clusters en la nube. El entorno de trabajo posee una interfaz muy similar a jupyter notebook, por lo que la adaptación del equipo al nuevo entorno de trabajo fue muy rápida. La subida de la información a la nube resulto algo más tediosa, pero el propio proveedor dio todo el soporte posible para facilitar la tarea. **Respecto a esto, Telefónica España estableció los acuerdos legales necesarios para poder subir sus datos con total seguridad a la nube de la empresa colaboradora. Lógicamente, la parte legal queda fuera de las responsabilidades del equipo de**

trabajo, pero fue un factor crítico a la hora de tener disponible los clusters on cloud.

- Un servidor on premise con 8 tarjetas gráficas incorporadas, al que se accede por un túnel SSH desde una máquina frontera.

Ambos despliegues emplean una de las gráficas más potentes, especialmente optimizada para el manejo de tensores en tareas de aprendizaje automático. El servidor on premise estuvo disponible en la etapa final del proyecto, por lo que se emplea principalmente el despliegue en la nube.

Definido el hardware necesario para implementar los modelos, se pasa a evaluar las distintas opciones de estos consideradas.

4.5.2 CONVOLUTIONAL NEURAL NETWORKS (CNN)

Las redes neuronales convolucionales (CNN, por sus siglas en inglés) son la estructura fundamental en el ámbito del aprendizaje profundo, Aunque su principal campo de aplicación se encuentra en el reconocimiento y clasificación de imágenes, también es posible aplicar esta estructura a problemas de procesamiento del lenguaje natural.

Independientemente del caso de aplicación, las redes convolucionales se basan en una sucesión de capas con filtros aplicadas sobre una matriz o conjuntos de matrices que representan el objeto de estudio (imagen, oración, etc.). Este conjunto de capas se entrena en la extracción de características del objeto. Habitualmente se emplean tres tipos de capas:

- Convolucional: Elemento encargado de la extracción de características. Se basa en el análisis de conjuntos de datos próximos para entender las relaciones entre éstos.
- ReLu: Esta capa introduce no linealidad en la red.
- Pooling: Capa centrada en reducir la dimensionalidad del problema.
- Capa completamente conectada (*fully connected layer)*: Suele ser la capa final en problemas de clasificación. Se centra en la clasificación de un objeto en una clase u otra en base a las características extraídas por las capas previas.

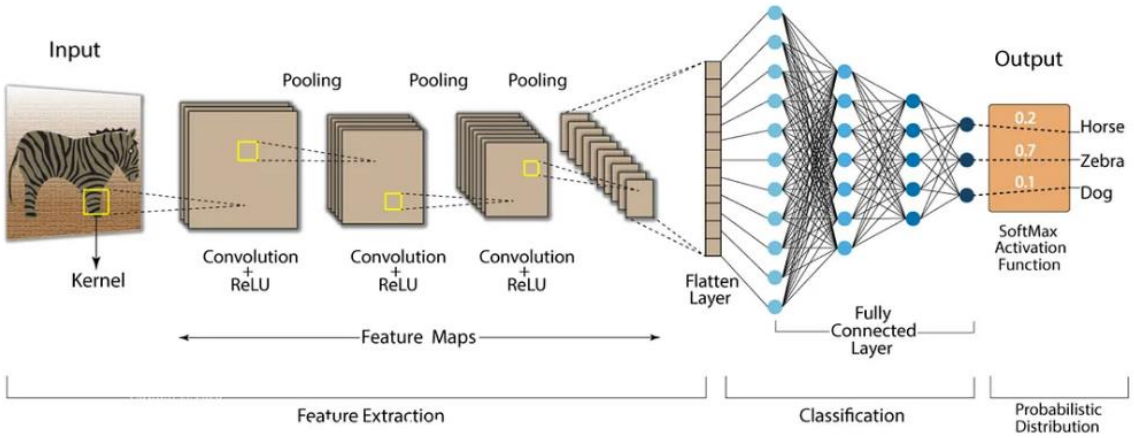

Figura 21. Arquitectura de las redes CNN (Fuente: What is Convolutional Neural Network — CNN (Deep Learning) | by Kh. Nafizul Haque | Medium)

Las capas ReLu, pooling y, principalmente, convolucional están centradas en la extracción de características de los inputs. En el caso de problemas de NLP el input consiste en una matriz que representa una oración, representando cada palabra de la misma con una fila de la matriz.

Durante el ajuste del modelo, los pesos de los filtros empleados en la red se van ajustando para maximizar la capacidad de clasificación de ésta, siguiendo una lógica similar a la empleada en el análisis de imágenes (Figura 22).

Uno de los grandes problemas de las redes CNN en relación al tratamiento de textos es que esta arquitectura analiza cada input de forma independiente respecto a los previos. Esto supone una limitación crítica en el procesamiento del lenguaje natural, porque no tiene sentido analizar una oración sin tener en cuenta las previas. Por ello, se deben buscar estructuras alternativas. En caso de tener más interés en las arquitecturas de redes CNN, existe mucha literatura al respecto (8) (9), pero su descripción en detalle excede los propósitos del proyecto y de este documento.

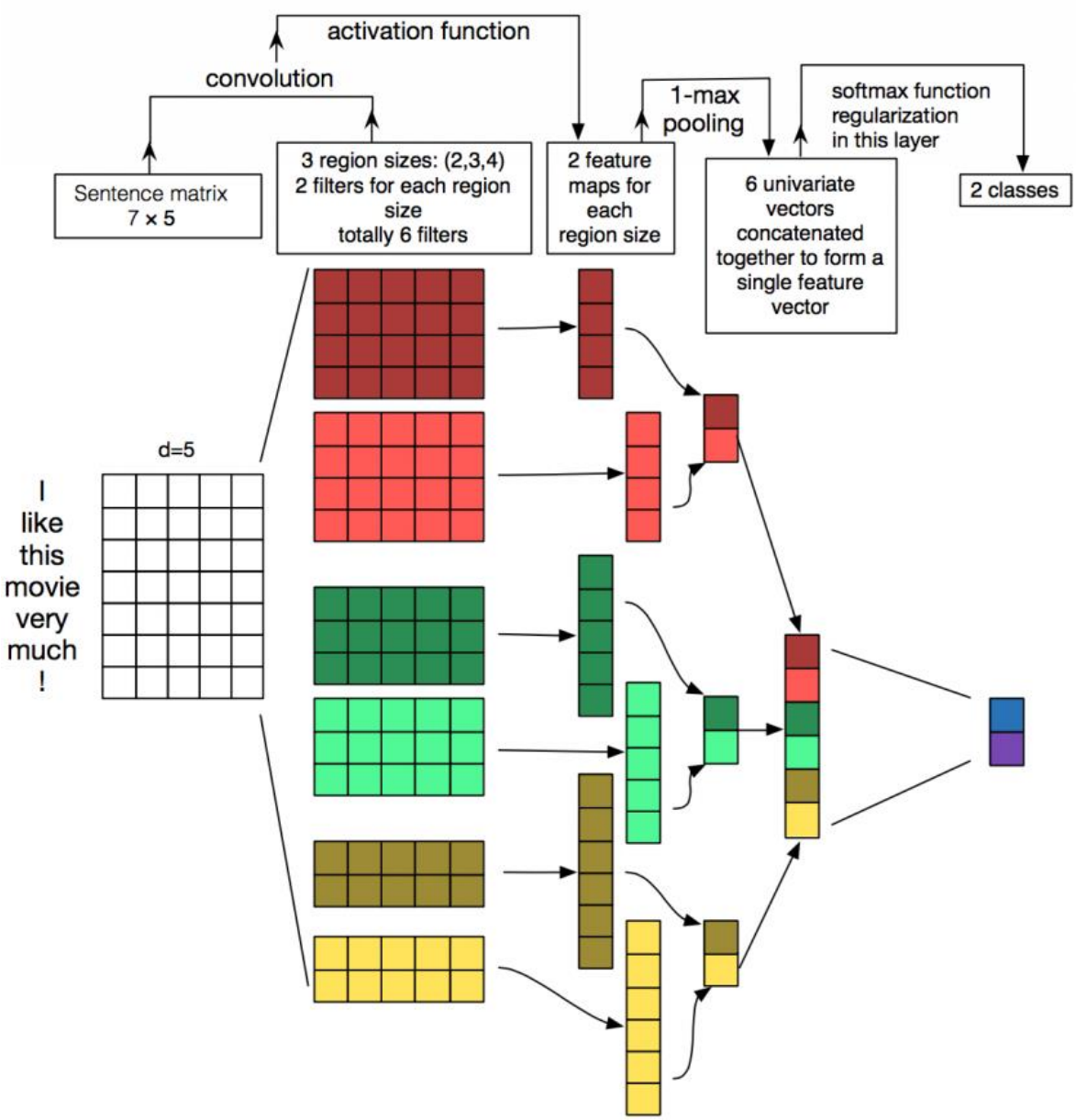

Figura 22. Flujo de una red convolucional aplicada al tratamiento de texto (Fuente: http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/)

4.5.3 RECURRENT NEURAL NETWORKS (RNN)

Las redes neuronales recurrentes (RNN, por sus siglas en inglés) mejoran los resultados obtenidos en las redes CNN al ser capaces de mantener información de inputs previos. Se denominan recurrentes porque realizan la misma tarea sobre cada elemento de una secuencia. A nivel conceptual pueden verse como un conjunto de copias de la misma red, en la que cada réplica de la red tiene en cuenta la salida de las anteriores (Figura 23D).

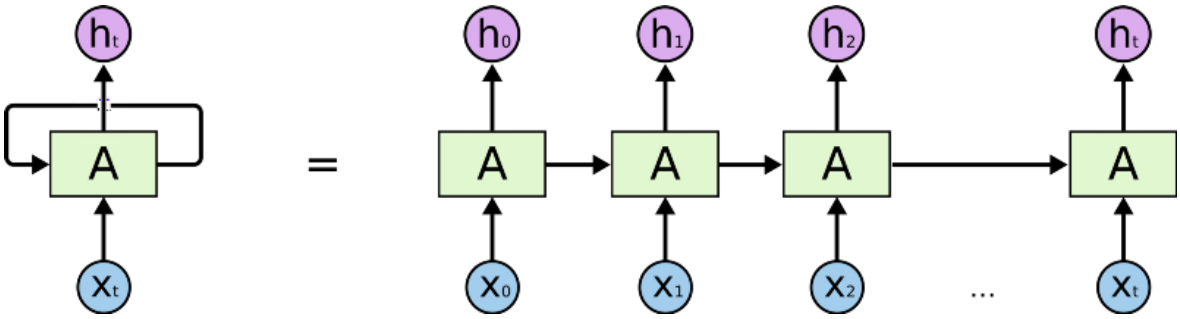

Figura 23. Esquema de una red neuronal recurrente. (Fuente: http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

De esta forma, la red mantiene una "memoria" de los elementos previos, lo que resulta muy útil en la modelización de series temporales o en el análisis de texto, donde es necesario tener en cuenta las frases previas para contextualizar las actuales. Desgraciadamente, se ha demostrado que estas redes sólo son efectivas manteniendo relaciones a corto plazo. En caso de querer relacionar partes de un texto muy distanciadas entre sí, las RNN no consiguen buenos resultados (10).

4.5.4 LONG SHORT-TERM MEMORY NETWORKS (LSTM)

Este tipo de red, denominadas LSTM por sus siglas en inglés, son capaces de manejar dependencias a largo plazo, resolviendo el principal problema de las RNN (de hecho, se trata de una variante de éstas).

Introducidas a finales de los 90 (11), han sido ampliamente utilizadas en los últimos años gracias a su "memoria" que permite mantener dependencias entre partes de un mismo texto muy distantes entre sí en el tiempo.

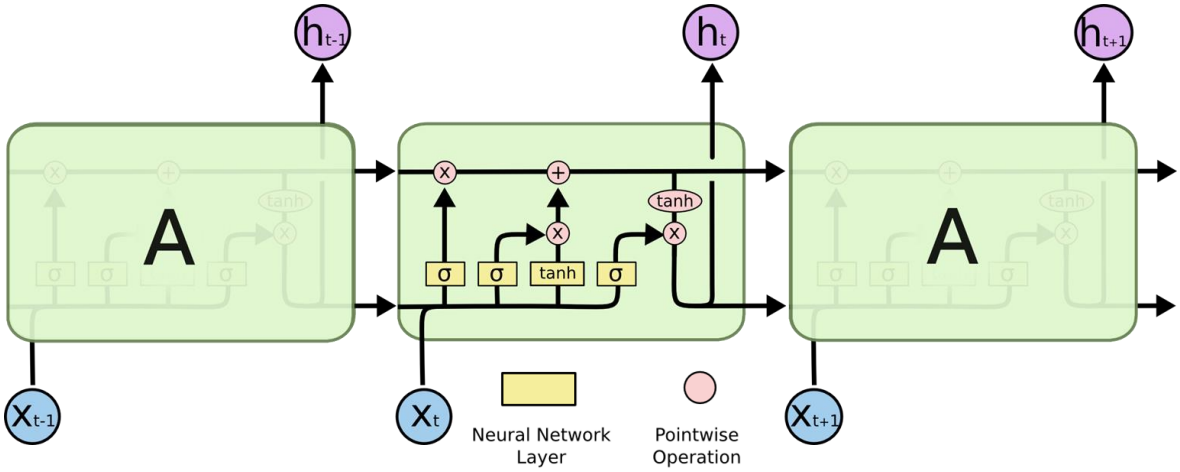

Figura 24. Esquema de una red LSTM. (Fuente: http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

Como se puede ver en la Figura 24, la red consta de cuatro redes neuronales. Estas redes se caracterizan por ser capaces de seleccionar qué información es relevante y cuál se puede eliminar. Para ello, la llamada "forget gate layer" (Figura 25) analiza la salida de la red previa (h_{t-1}) y el nuevo input (x_t), y decide qué información persiste y cual se elimina.

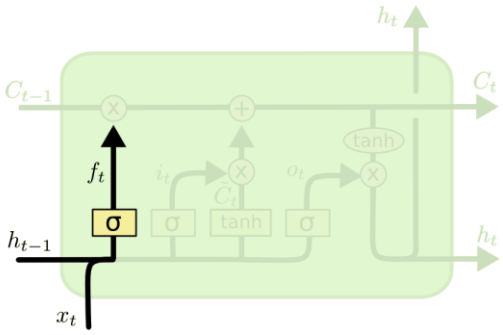

Figura 25. Detalle de la "forget gate layer" de una red LSTM

(Fuente: http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

Eliminada información irrelevante, la red decide que parte de la información nueva se va a persistir en la memoria de esta. Para ello emplea la llamada "input gate layer" que usa una capa sigmoide. A continuación, una capa de tangente hiperbólica establece el valor de la nueva información que la "input gate layer" ha decidido persistir en memoria (Figura 26)

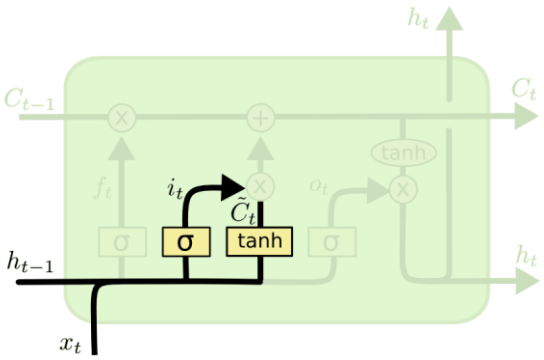

Figura 26. Detalle de la "input gate layer" y la capa con tanh en la red LSTM. (Fuente: http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

Una vez se ha definido qué información se va a mantener en memoria y su valor, se actualiza el estado previo con dicha información (Figura 27).

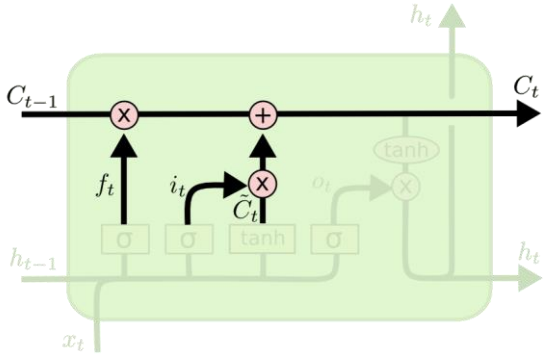

Figura 27. Actualización de la información que la red transfiere hacia etapas posteriores. (Fuente: http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

La última red neuronal es la encargada de generar el output de la red LSTM. Consta de una sigmoide, que filtra la información de salida, y una tangente hiperbólica que normaliza su valor entre -1 y 1 (Figura 28).

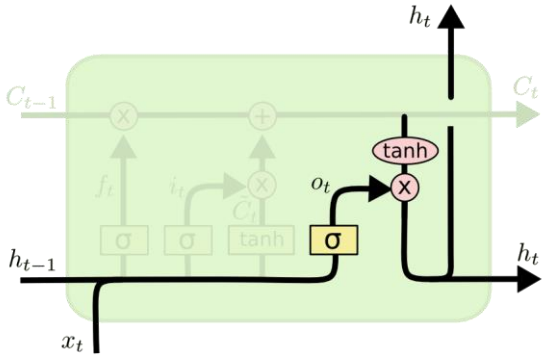

Figura 28. Definición de la salida de una red LSTM. (Fuente: http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

Una de las variantes empleadas en el proyecto fueron las redes LSTM bidireccionales. Además de tener en cuenta los eventos pasados en los inputs actuales, estas redes también tienen en cuenta los eventos futuros, de ahí que se denominen bidireccionales.

4.5.5 GATED RECURRENT UNITS (GRUS)

Este tipo de redes (GRU, por sus siglas en inglés) constituyen una de las variantes más populares de las LSTM. Introducidas en 2014 (12), este tipo de red emplea únicamente dos tipos de puertas:

- "Update gate": determina qué información del pasado se debería mantener en el estado de la red.
- "Reset gate": determina qué información es irrelevante y se debe eliminar del estado.

4.5.6 AJUSTE DE MODELOS Y RESULTADOS

En base a lo explicado en los apartados previos, inicialmente se apuesta por las redes LSTM, pues parecen ser las más prometedoras dentro de las estructuras de deep learning.

Para el ajuste se divide el set de datos en dos partes: el 80% se emplea en entrenamiento y el 20% restante en la validación del modelo ajustado.

Una de las dificultades detectadas en esta fase es el desbalanceamiento del set de datos. En el conjunto existe un volumen mayor de clientes satisfechos que insatisfechos, por lo que un algoritmo obtendrá buenos resultados simplemente al precedir como más probable el colectivo mayoritario. Para resolver el problema del desbalanceo se opta repetir aleatoriamente los registros de la clase minoritaria. Como se aprecia en la Figura 29, se realiza fácilmente con la librería ibmlearn:

```
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(random_state=42)
x_train_rs, y_train_rs = ros.fit_resample(x_train, y_train)
```

Figura 29. Remuestreo del conjunto de entrenamiento (x_train e y_train).

A continuación se lanzan las pruebas empleando una GPU. En estas condiciones las redes LSTM requieren entre una y tres horas por cada época, y emplean entre 10 y 30 épocas por cada modelo.

Tras varias pruebas, la estructura más conveniente emplea una red LSTM bidireccional, aunque también incluye una capa convolucional. Los resultados obtenidos, presentados en la Tabla 7, muestran unos resultados razonablemente prometedores. Aunque la accuracy es menor a la obtenida con fastText, y existe cierto sobreentrenamiento, el valor de recall es mejor al obtenido empleando la librería de Meta.

Modelo (Red)	Accuracy entrenamiento	Accuracy test	Precision	Recall
BiLSTM + CNN	0,7335	0,6875	0,27	0,65

Tabla 6. Resultados obtenidos con una de las primeras redes LSTM

No obstante, el modelo presenta una tendencia clara al sobreentrenamiento (Figura 30), especialmente a partir de la tercera época (los resultados de la Tabla 7 se obtienen en la primera época, es decir, con el primer checkpoint). Además, la accuracy en validación presenta valores dispares, lo que hace dudar de la validez del modelo. Conforme avanzan las épocas, los resultados en precisión y recall tendían a empeorar. Dados los elevados tiempos de entrenamiento necesarios, no es viable realizar validación cruzada.

```
Epoch 1/10
 - 1876s - loss: 0.5377 - acc: 0.7335 - val_loss: 0.6347 - val_acc: 0.6875
Epoch 2/10
 - 1886s - loss: 0.3465 - acc: 0.8520 - val_loss: 0.5120 - val_acc: 0.8155
Epoch 3/10
 - 1888s - loss: 0.1671 - acc: 0.9364 - val_loss: 0.7868 - val_acc: 0.7953
Epoch 4/10
 - 1887s - loss: 0.0771 - acc: 0.9728 - val_loss: 1.0054 - val_acc: 0.8258
Epoch 5/10
 - 1885s - loss: 0.0410 - acc: 0.9863 - val_loss: 1.1229 - val_acc: 0.8023
Epoch 6/10
 - 1887s - loss: 0.0277 - acc: 0.9907 - val_loss: 1.2936 - val_acc: 0.8182
Epoch 7/10
 - 1871s - loss: 0.0197 - acc: 0.9935 - val_loss: 1.4450 - val_acc: 0.8180
```

Figura 30. *Accuracy y función de pérdida, en entrenamiento y validación, para el modelo con red LSTM.*

En la Figura 31 se puede ver una de las virtudes y, a su vez, defectos de este tipo de modelos. Resulta muy sencillo incorporar diferentes tipos de capas en la estructura, pero cada una suele incluir un elevado número de hiperparámetros que son susceptibles de requerir un ajuste. Esto dificulta mucho la obtención de los hiperparámetros óptimos.

```
model=Sequential();
embedding_layer = Embedding(num_words,
                            EMBEDDING_DIM,
                            embeddings_initializer=Constant(embedding_matrix),
                            input_length=MAX_SEQUENCE_LENGTH,
                            trainable=True)
model.add(embedding_layer)
model.add(Bidirectional(LSTM(units=64, return_sequences=True)))
model.add(Conv1D(filters=32, kernel_size=3, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(1, activation='sigmoid'))
optimizer = keras.optimizers.Adam()
```

Figura 31. Elementos del modelo con red LSTM bidireccional ajustado en el cluster en la nube.

Precisamente para intentar resolver la cuestión del ajuste de hiperparámetros, se decide emplear Optuna. Optuna, al igual que en fastText, permite realizar un conjunto de pruebas

en las que se van variando los hiperparámetros. En cada ensayo se asignan a éstos los valores que son susceptibles de conseguir unos resultados óptimos.

A diferencia de en fastText, no es posible lanzar en paralelo diferentes pruebas, de modo que se reduzca el tiempo de ejecución. La razón se debe a que sería necesario una base de datos PostgreSQL que no resulta sencilla de configurar en el entorno cloud. Sí es posible lanzar el proceso de optimización de parámetros, de modo que Optuna lanza pruebas sucesivas que, en teoría, obtienen mejores resultados. Con esta estrategia se consiguen dos modelos con resultados bastante positivos.

Para aumentar la fiabilidad de los modelos, se decide dividir la muestra en tres partes:

- En entrenamiento se emplean el 80% de los datos disponibles (set de entrenamiento).
- Un 10% se reserva para la evaluación (set de validación) que se realiza en el desarrollo del ajuste del modelo, de modo que se van optimizando los hiperparámetros.
- El 10% restante se usa como set de datos completamente nuevo para el modelo (set de test). Si la calidad del modelo ajustado es suficientemente alta, los resultados con el set de validación y test deberían ser similares.

Además, en la fase de entrenamiento con Optuna no se replicaron datos para mantener el balanceo entre clases.

En el primero modelo ajustado con Optuna se emplea una red CNN bastante sencilla por lo que las diversas pruebas se realizan en tiempos razonables. Finalmente, los mejores hiperparámetros son los que se muestran en la Figura 32.

```
model = Sequential()
model.add(Embedding(input_dim=32536, output_dim=300, input_length=2723))
model.add(Conv1D(filters=54, kernel_size=4, activation='relu'))
model.add(MaxPooling1D(pool_size=3))
model.add(Flatten())
model.add(Dropout(rate=0.4977))
model.add(Dense(1, activation='sigmoid'))
optimizer = keras.optimizers.Adam()
```

Figura 32. Estructura del modelo CNN ajustado mediante Optuna.

A continuación, se decide modificar ligeramente la estructura, introduciendo una red GRU (Apartado 4.5.5) ya que se espera que los resultados pudieran mejorar al ser un tipo de red que maneja mejor las dependencias entre elementos de un texto que las CNN, como se ha explicado. El esquema del mejor modelo conseguido con Optuna se aprecia en la Figura 33.

```
model = Sequential()
model.add(Embedding(input_dim=32536, output_dim=300, input_length=2723))
model.add(Conv1D(filters=183, kernel_size=5, activation='relu'))
model.add(SpatialDropout1D(rate=0.7577))
model.add(MaxPooling1D(pool_size=5)
model.add(CuDNNGRU(units=180))
model.add(Dropout(rate=0.876224))
model.add(Dense(1, activation='sigmoid'))
```

Figura 33. Estructura del segundo modelo ajustado mediante Optuna, combinando CNN y GRU.

Los resultados de ambos modelos se resumen en la Tabla 7. Como se aprecia, no hay una diferencia destacada entre ambos modelos, aunque, paradójicamente, el más sencillo obtiene valores mayores de precisión y, particularmente, recall. Por tanto, entre ambos, parece ser mejor optar por el modelo más simple, sólo con red CNN

Modelo (Red)	Accuracy entrenamiento	Accuracy validación	Accuracy test	Precision	Recall
CNN	0.9210	0.8298	0.8312	0.378	0.279
CNN + GRU	0.9032	0.8149	0.8188	0.304	0.206

Tabla 7. Resultados de las redes entrenadas con Optuna.

A luz de los resultados de la Tabla 6 y Tabla 7, resulta razonable realizar un barrido de parámetros con Optuna empleando una estructura con una red LSTM combinada con CNN, ya que este tipo de red es la que consigue mejores resultados de recall. La ejecución de una prueba así requeriría tiempos de computación muy elevados. Es lógicamente viable realizar esta prueba, pero no ha sido posible dentro de los plazos del proyecto pues existe un límite en el uso de la plataforma cloud. En fases sucesivas se sugiere evaluar dicha configuración, pues existen experiencias que demuestran que esta configuración puede alcanzar buenos resultados (13).

En este sentido, durante la recta final del proyecto el equipo recibió una máquina con 8 GPUs de última generación. En esta máquina sí es posible instalar una base de datos PostgreSQL similar a la empleada con fastText. El equipo del proyecto comenzó con las pruebas con Optuna, con ensayos en paralelo, pero no fue posible alcanzar resultados dentro del plazo del Trabajo Fin de Master.

4.6 BERT

Otra de las tecnologías consideradas en el desarrollo del proyecto fueron las técnicas de "transfer learning", basadas en el uso de modelos preentrenados a los que se le aplica un ajuste fino para adaptarlos al problema del caso de uso.

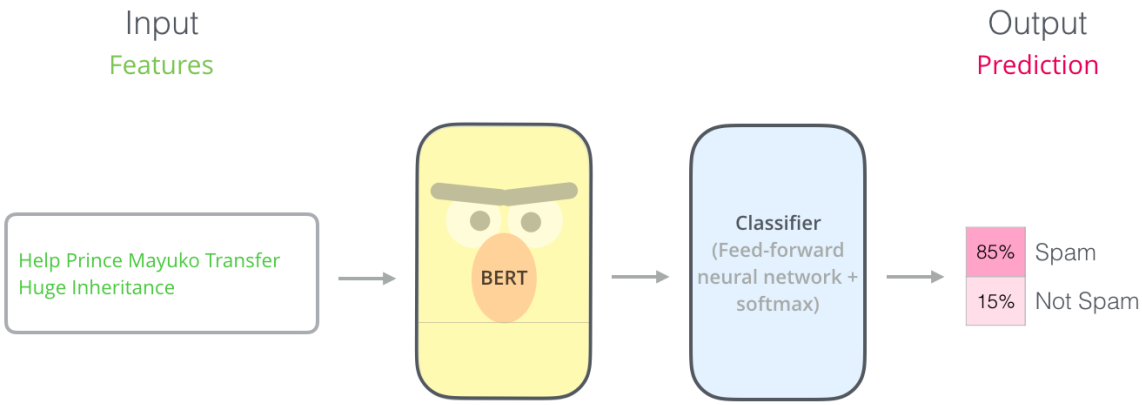

Figura 34. Esquema básico del caso de uso de clasificación de oraciones de BERT. (Fuente: http://jalammar.github.io/illustrated-bert/)

De entre las herramientas más potentes destaca BERT (Bidirectional Encoder Representations from Transformers), desarrollado por Google (1). BERT es una herramienta de NLP que permite extraer patrones en el lenguaje. El modelo ha sido preentrenado a partir de un conjunto de corpus. Posteriormente, dicho modelo puede ser empleado en modelos de clasificación, como la clasificación de oraciones (Figura 34).

En cuanto a la arquitectura, el paper de BERT (1) incluye dos versiones del modelo: el modelo BASE, de tamaño más reducido, y el modelo LARGE, considerablemente mayor.

En ambos casos el modelo consiste en un conjunto de encoders ("Transfer blocks" en el paper) encargados de procesar cada input (en el caso de uso, cada conversación) y devolver una salida que se pasa por un clasificador (Figura 35).

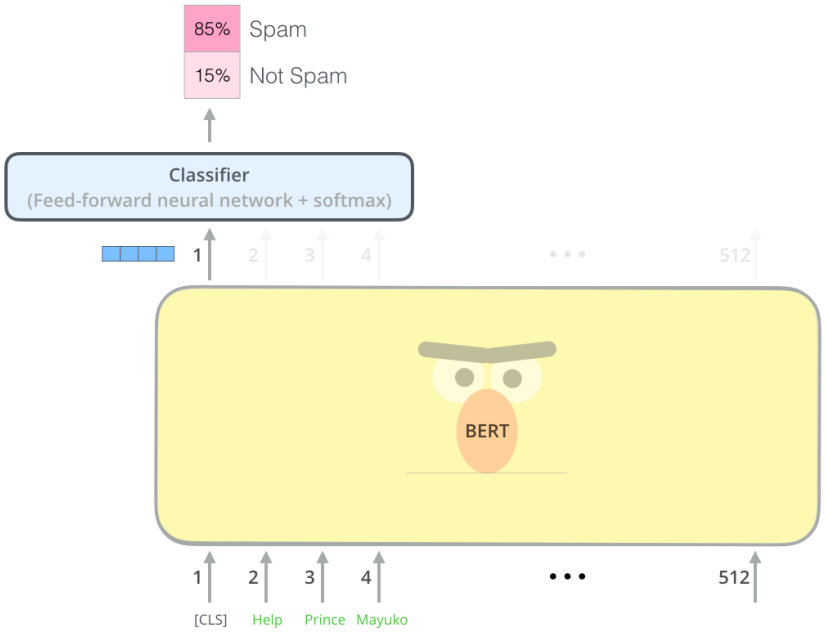

Figura 35. Esquema general del procesamiento de una oración por parte de BERT. (Fuente: http://jalammar.github.io/illustrated-bert/)

Conocer la arquitectura en profundidad de la herramienta requiere conocer conceptos previos como el "Transformer". Su descripción detallada excede el alcance de este documento. En la bibliografía del proyecto es posible encontrar explicaciones en detalle de la misma (1) (14) (15).

4.6.1 AJUSTE DE MODELOS Y RESULTADOS

En la práctica el ajuste fino de BERT resulta moderadamente sencillo ya que existen pocos hiperparámetros a definir. Una de las ventajas de la herramienta es que dispone de varios corpus en diferentes idiomas, incluido el español.

Uno de los hándicaps de BERT es que permite un tamaño máximo de oración de 512 tokens, mientras que las conversaciones del proyecto tienen longitudes de 1000 palabras de media. Existen, no obstante, herramientas específicas para solventar este problema. Longformer y BigBird, por ejemplo, permiten incrementar las entradas hasta en 8.000 tokens. Aunque BERT se considera una de las herramientas más punteras en cuanto a las técnicas de NLP, los resultados obtenidos no son buenos. No se ha conseguido que el modelo se ajuste correctamente al set de datos y al caso de uso, no se obtiene un valor fiable de accuracy, precisión o recall. En concreto, el problema parece ser la adaptación del framework al idioma español (lógicamente se ha desarrollado originalmente en inglés).

4.7 MÁS ALLÁ DE BERT: ROBERTA Y DEBERTA

Con el objetivo de mejorar la capacidad predictiva del sistema desarrollado, se ha considerado la incorporación de algoritmos de reciente aparición que han demostrado un desempeño destacado en tareas de procesamiento de lenguaje natural, especialmente en clasificación de sentimientos y análisis de intención. Entre los modelos evaluados destacan variantes avanzadas de la arquitectura Transformers, como **RoBERTa** (2019) y **DeBERTa** (2021), los cuales introducen optimizaciones sustanciales con respecto al modelo BERT original.

RoBERTa (Robustly Optimized BERT Approach) elimina el uso de *segment embeddings* y aumenta el tamaño del corpus y la duración del entrenamiento, lo que incrementa la capacidad del modelo para capturar matices semánticos complejos. En concreto, ha sido entrenado con 160gb de texto, más de 10 veces el tamaño del dataset empleado para entrenar BERT.

Por su parte, **DeBERTa** (Decoding-enhanced BERT with disentangled attention) (16) introduce mecanismos de atención desentrelazada y un decodificador de enmascaramiento mejorado (*enhanced mask decoder*), lo que mejora su comprensión del contexto textual.

El mecanismo de atención desentrelazada representa cada palabra utilizando dos vectores separados que codifican, respectivamente, su contenido y su posición (Figura 36). Los pesos de atención entre palabras se calculan utilizando matrices desentrelazadas aplicadas a sus contenidos y posiciones relativas.

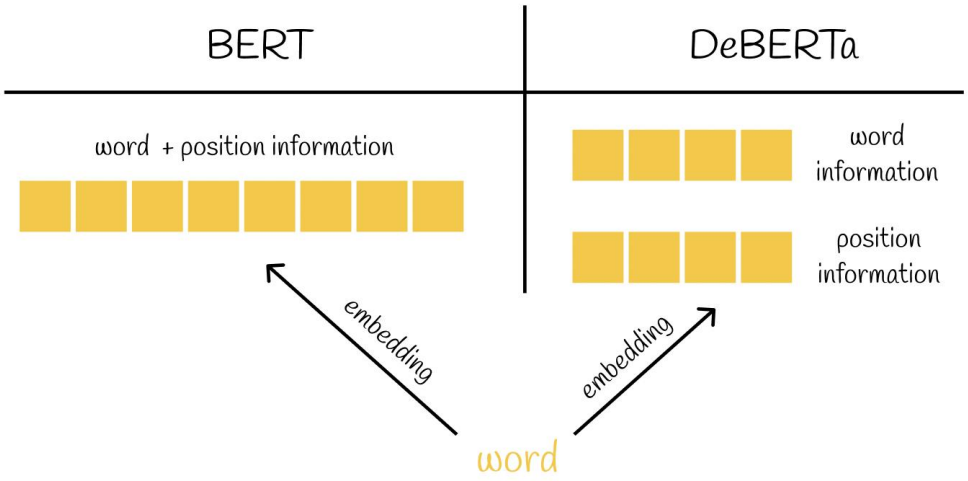

Figura 36. Construcción de embedding en BERT y DeBERTa. (Fuente: Large Language Models: DeBERTa - Decoding-Enhanced BERT with Disentangled Attention | Towards Data Science)

El decodificador de enmascaramiento mejorado (Figura 37) se emplea en lugar de la capa *softmax* de salida tradicional para predecir los tokens enmascarados durante la fase de preentrenamiento del modelo.

No obstante, un desafío central en la aplicación de estos modelos en nuestro caso particular es su adaptación al idioma español. La mayoría de las implementaciones originales de estos algoritmos están preentrenadas en inglés, lo que limita su capacidad para capturar con precisión las particularidades lingüísticas y culturales de las conversaciones analizadas. Si bien existen versiones multilingües (por ejemplo, mBERT o XLM-R), estas tienden a tener un rendimiento inferior al de los modelos monolingües entrenados específicamente para un idioma.

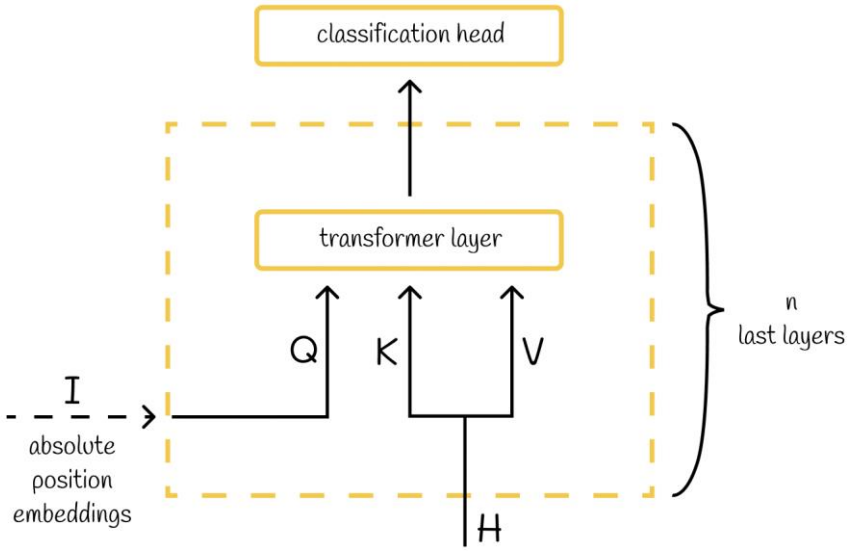

Figura 37. Decodificador de enmascaramiento mejorado (Fuente: Large Language Models: DeBERTa - Decoding-Enhanced BERT with Disentangled Attention | Towards Data Science)

En esta línea, se plantea como mejora futura la utilización de modelos preentrenados en corpus amplios de español, como **BETO** o variantes en español de RoBERTa y DeBERTa, combinadas con técnicas de *fine-tuning* sobre el dominio específico de atención al cliente. También se valorará el uso de técnicas de *data augmentation* lingüísticamente informadas para enriquecer el corpus de entrenamiento.

Aunque se realizaron pruebas preliminares con algunas de estas variantes, los resultados no fueron concluyentes. Se considera que un ajuste más fino al dominio conversacional en español podría permitir mejoras significativas en la predicción del grado de satisfacción, habilitando intervenciones más precisas en tiempo real por parte de la empresa.

Capítulo 5. CONCLUSIONES

En el desarrollo del proyecto se ha evaluado el uso de técnicas de procesamiento del lenguaje natural para predecir el grado de satisfacción de los consumidores en su contacto telefónico con Telefónica España. Conocida la satisfacción de un cliente, la empresa podría tomar acciones pertinentes para evitar su baja.

Durante el proyecto también se ha realizado una comparativa entre diferentes sistemas de transcripción. Si bien la selección de la herramienta de transcripción excede las responsabilidades del equipo de trabajo, se puede demostrar que el servicio 2 obtiene unos resultados estadísticamente peores.

Obtenido el texto, se desarrollan tres tipos de modelos de machine learning:

- fastText: Librería open source desarrollada por Meta. Emplea redes poco profundas y está escrito en C++, por lo que es muy rápido. Ha sido el primer modelo probado gracias a su rapidez y simplicidad en el despliegue, pues está optimizado para CPUs, sin ser necesario emplear GPUs.
- Redes convolucionales y recurrentes: Creados en Keras bajo el entorno de jupyter notebook en Python, han conseguido resultados bastante positivos, aunque tal vez peores de los esperados. En función del tipo de red, los tiempos de entrenamiento pueden ser largos, lo que se podría solventar empleando herramientas como Optuna para la optimización de hiperparámetros.
- Modelos de transfer learning. Concretamente se ha empleado BERT, una de las herramientas más completas. Desgraciadamente, no ha sido posible obtener resultados concluyentes, por lo que se plantean alternativas a BERT que podrían dar lugar a mejores resultados. La más prometedora de ellas es RoBERTa, que introduce mejoras relevantes con respecto a BERT y ha conseguido mejores resultados en diversos benchmark. Más allá de la arquitectura empleada, resulta crítico adaptar el

algoritmos seleccionado a la lengua castellana, por lo que alternativas como BETO pueden dar lugar a buenos resultados.

En la Tabla 8 se muestran los resultados obtenidos con los diferentes modelos creados en el desarrollo del proyecto. Como conclusiones relevantes se destacan:

- Los resultados obtenidos con fastText son bastante buenos, pese a no emplear GPUs.
- A la hora de maximizar precisión y recall, las redes LSTM sí parece que den mejor resultados que las CNN o CNN + GRU, pero los tiempos de entrenamiento son considerablemente mayores, lo que dificulta la realización de múltiples pruebas con Optuna para fijar hiperparámetros.
- Respecto a la accuracy, las redes CNN obtienen unos resultados bastante positivos, aunque se tiende al sobreentrenamiento en mayor medida que en las redes LSTM. Los tiempos de entrenamiento con CNN son mucho más cortos, lo que facilita realizar múltiples pruebas con distintas configuraciones y, por tanto, la optimización de hiperparámetros.

Modelo (Red)	Uso de Optuna	Accuracy entrenamiento	Accuracy validación	Accuracy test	Precision	Recall
fastText	Sí	0.8596	-	0.8392	0.59	0.12
BiLSTM + CNN	No	0.7335	-	0.6875	0.27	0.65
CNN	Sí	0.9210	0.8298	0.8312	0.38	0.28
CNN + GRU	Sí	0.9032	0.8149	0.8188	0.30	0.21

Tabla 8. Resumen de los resultados obtenidos con todos los modelos creados durante el proyecto.

En etapas sucesivas del proyecto se sugiere desarrollar pruebas en paralelo con Optuna, como las empleadas en fastText, en modelos con redes LSTM bidireccionales, que parecen ser los más positivos en recall.

Finalmente, ya que en todo problema de analítica el tamaño del set de datos es muy relevante, se recomienda seguir aumentando la muestra de llamadas descargadas y transcritas.

Capítulo 6. REFERENCIAS

1. **Devlin, Jacob, y otros.** *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* 2018. 1810.04805.

2. **Mikolov, Tomas, y otros.** *Efficient Estimation of Word Representations in Vector Space.* 1301.3781v3.

3. **Huang, Steeve.** Word2Vec and FastText Word Embedding with Gensim – Towards Data Science. [En línea] 2018. https://towardsdatascience.com/word-embedding-with-word2vec-and-fasttext-a209c1d3e12c.

4. **Bojanowski, Piotr, y otros.** *Enriching Word Vectors with Subword Information.* 1607.04606v2.

5. **Joulin, Armand, y otros.** *Bag of Tricks for Efficient Text Classification.* págs. 427-431.

6. **Subedi, Nishan.** FastText: Under the Hood – Towards Data Science. [En línea] 2018. https://towardsdatascience.com/fasttext-under-the-hood-11efc57b2b3.

7. **Bojanowski, Piotr.** NLP Meetup #2 - fastText (by Piotr Bojanowski) - YouTube. [En línea] 2016. https://www.youtube.com/watch?v=CHcExDsDeHU.

8. **Karn, Ujjwal.** An Intuitive Explanation of Convolutional Neural Networks - the data science blog. [En línea] 2016. https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/.

9. **Mishra, Mayank.** Convolutional Neural Networks, Explained. [En línea] 2019. https://www.datascience.com/blog/convolutional-neural-network.

10. **Bengio, Yoshua, Simard, Patrice y Frasconi, Paolo.** *Learning Long-Term Dependencies with Gradient Descent is Difficult.* 1994.

11. **Hochreiter, Sepp y Urgen Schmidhuber, J J.** *Long Short-Term Memory.* 1997. págs. 1735-1780.

12. **Cho, Kyunghyun, y otros.** *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation.* 1406.1078v3.

13. **M. Sosa, Pedro.** Twitter Sentiment Analysis using combined LSTM-CNN Models – B-sides. [En línea] 2018. http://konukoii.com/blog/2018/02/19/twitter-sentiment-analysis-using-combined-lstm-cnn-models/.

14. **Alammar, Jay.** The Illustrated Transformer – Visualizing machine learning one concept at a time. [En línea] 2018. https://jalammar.github.io/illustrated-transformer/.

15. The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning) – Visualizing machine learning one concept at a time. [En línea] 2018. http://jalammar.github.io/illustrated-bert/.

16. **Pengcheng He, Xiaodong Liu, Jianfeng Gao, Weizhu Chen.** *DeBERTa: Decoding-enhanced BERT with Disentangled Attention.* 2020. [2006.03654v6] DeBERTa: Decoding-enhanced BERT with Disentangled Attention