Comparativa de técnicas de análisis de sentimiento en contextos competitivos
Comparison of sentiment analysis techniques in competitive contexts

Trabajo de Fin de Máster
Curso 2020–2021

Autor
Raquel Blanco Morago

Director
Rafael Caballero Roldán

Convocatoria: *junio-julio*
Calificación: *9*

Resumen

Comparativa de técnicas de análisis de sentimiento en contextos competitivos

Cada vez es más común utilizar sistemas automáticos como *IBM Watson* para tareas tales como detectar la opinión en textos, el llamado *análisis de sentimiento*. Estos sistemas reemplazan a los modelos específicos en los que se seleccionaba un subconjunto de textos que se etiquetaban manualmente para luego generar un modelo para cada corpus concreto.

En este trabajo mostramos experimentalmente que, a pesar de los avances logrados por estos sistemas automáticos, la generación de modelos con entrenamiento manual sigue resultando útil en ciertas situaciones. Para ello hemos elegido textos de dos tipos de ejemplos: un primer tipo al que llamamos de *contexto competitivo* que incluye comentarios en redes sociales acerca de acontecimientos como procesos electorales, y un segundo tipo que incluye reseñas online acerca de productos o servicios. Nuestros resultados concluyen que incluso un método de clasificación sencillo basado en el entrenamiento manual puede superar la eficiencia de *IBM Watson* cuando se consideran ejemplos del primer tipo, mientras que *IBM Watson* sí supera a este método entrenado manualmente en los casos del segundo tipo.

Además, analizamos varios factores que pueden afectar a los resultados de estos experimentos, descartando la influencia en nuestros ejemplos de aspectos tales como la complejidad del texto o los errores ortográficos. También examinamos el papel de elementos específicos de las redes sociales y en particular de los *tweets*, tales como *hashtags*, menciones o enlaces externos, encontrando que estos elementos sirven para transmitir al menos parte del contexto que Watson no puede detectar.

Como contribución adicional que también merece ser comentada, presentamos una nueva métrica basada en la popular Kappa de Cohen a la que hemos llamado Kappa Penalizada. Aunque ya existen variantes de Kappa que buscan penalizar el resultado de Kappa teniendo en cuenta la distancia entre el valor predicho y el real, hemos encontrado que estas propuestas tienen un comportamiento anómalo al ser sus valores a menudo mayores que el valor de la Kappa original. Nuestra propuesta reduce significativamente el número de veces que esto sucede.

Palabras clave

Análisis de sentimiento, CRISP-DM, IBM Watson, Kappa de Cohen, Twitter

Abstract

Comparison of sentiment analysis techniques in competitive contexts

Today it is increasingly common to use automatic systems like *IBM Watson* in order to accomplish tasks like text opinion detection, also called *sentiment analysis*. These systems replace the specific models where we used to select a subset of texts that were manually labeled for the purpose of generating one model for each particular corpus.

In this project we experimentally show that, despite the breakthroughs reached by these automatic systems, model generations with manual training are still useful in some situations. In order to achieve this goal we have chosen two types of examples: the first one that we called *competitive context* includes social networks commentaries about events like electoral processes, and the second one that includes online reviews about products or services.

Our results conclude that even a simple classification method based on manual training can overcome the performance of *IBM Watson* when examples of the first type are considered, whereas *IBM Watson* outperforms this manual training method in the second type cases.

In addition, we analyzed several factors that can affect the results of these experiments, rejecting the influence of features like text complexity or misspellings in our examples. We also examine the role of specific elements in social networks, especially in tweets, such as hashtags, mentions and external links, discovering that these elements are useful to transmit at least part of the context that Watson cannot detect.

As an additional contribution that also deserves to be mentioned, we introduce a new metric based on the well-known metric Cohen's Kappa named Penalized Kappa. Although there are some Kappa variations that already seek to penalize taking into account the distance between the predicted and the real value, we have found that these proposals can show an anomalous behaviour because their values often are higher than the value of the original Kappa. Our proposal reduces significantly the number of times that these situations happen.

Keywords

Cohen's Kappa, CRISP-DM, IBM Watson, Sentiment analysis, Twitter

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este capítulo introductorio se comentará en primer lugar la motivación que nos ha llevado al desarrollo de este trabajo y se describirán los objetivos que este trabajo pretende resolver. A continuación se presentará el término "análisis de sentimiento" y algunas de las herramientas automáticas disponibles, y finalmente se describirá la estructura que sigue la memoria.

1.1. Motivación

En los últimos años se ha extendido el uso de herramientas automáticas o semiautomáticas para realizar labores de *machine learning* (aprendizaje automático) como *análisis de sentimiento*. Este es el caso de *IBM Watson* (Hig12), el sistema de la compañía IBM que, entre otras muchas características, incluye el análisis de sentimiento. La ventaja principal de estos sistemas es que nos ahorramos todo el trabajo que genera la elaboración de un sistema de *machine learning*, en particular la tediosa tarea de clasificar a mano en el caso de aprendizaje supervisado. El usuario simplemente envía el texto que se quiere analizar y el tema o *target* sobre el que se desea conocer la opinión transmitida por dicho texto, y el sistema devuelve esta opinión como negativa, positiva o neutra. Se trata, por tanto, de modelos previamente entrenados a partir de una gran cantidad de textos, lo que además debería hacer que obtuvieran resultados mejores que modelos específicos entrenados con unos pocos ejemplos.

Sin embargo, persiste la duda de si estos sistemas superan en general a los modelos específicos o si, por el contrario, existirán casos donde siga precisándose del entrenamiento manual para obtener mejores resultados. En particular, en este trabajo nos planteamos si en el caso de los mensajes emitidos por usuarios en redes sociales en relación con eventos competitivos tales como campañas electorales los modelos previamente entrenados de Watson pueden llegar a funcionar peor que en otros casos. La razón es que parece que, a priori, este tipo de mensajes, por un lado, tienen una fuerte dependencia contextual de noticias externas y, por otro, que dado el elevado número de mensajes y la intensidad de noticias se llega a desarrollar una especie de lenguaje propio, donde algunas palabras pueden llegar a tener un significado diferente al usual, que será el asumido por Watson.

Para ello nos proponemos comparar la eficiencia del sistema Watson con respecto a un modelo de aprendizaje supervisado básico desarrollado por nosotros y entrenado a partir de etiquetas generadas para ese problema específico en un conjunto de ejemplos.

1.2. Objetivos

El objetivo principal de este trabajo es encontrar respuesta para las dos siguientes preguntas:

- P1: ¿Hay ejemplos reales en los que Watson obtenga peores resultados que un clasificador específico entrenado manualmente?
- P2: Si hay diferencias ¿podemos detectar a qué se deben? ¿tienen influencia aspectos como el uso de elementos utilizados comúnmente en *Twitter* (*hashtags*, *URLs* embebidas...), las palabras mal escritas o la complejidad lingüística del texto?

Con respecto a la pregunta P1 sabemos que Watson puede proporcionar malos resultados por ejemplo si el idioma del texto de entrada no es el que espera, es decir si por ejemplo se le indica que el texto está escrito en inglés cuando en realidad el texto está escrito en castellano. Obviamente en este caso se trata de un engaño al sistema y no es lo que pretendemos, pero esto nos sugiere que algo similar podría ocurrir en un texto en inglés si las palabras se alejan del inglés estándar. Este "alejamiento" puede ser en principio de dos tipos:

1. Sintáctico, ya sea por palabras mal escritas o símbolos desconocidos para Watson.
2. Semántico, donde las palabras en apariencia son correctas pero tienen un significado diferente del usual, por ejemplo porque ciertas palabras clave relacionen para el lector el texto con una noticia externa. Si esto ocurre el entrenador humano tendrá en cuenta este contexto mientras que Watson funcionará peor por tener solo en cuenta el significado estándar. Otro problema que tiene Watson con respecto al contexto es que en los textos de tipo competitivo existe más de un *target*, y a menudo el mensaje se refiere a él indirectamente (ver Ejemplo 4.7.2) por lo que es fácil que Watson no logre comprender el sentimiento ya que le es difícil comprender a quién se refiere el texto, mientras que en los casos de las reseñas solo hay un *target*.

Obsérvese que pudiera ser que nos encontráramos con símbolos que no son interpretables para Watson (problema sintáctico) pero que para la persona que clasifica los textos sí tengan un significado en el contexto en el que se han emitido los mensajes, como por ejemplo puede ser la utilización de *hashtags* en *tweets*, cuya relevancia, en particular en el contexto de temas relacionados con la política, es bien conocida (véase por ejemplo (ASNH17)).

Fijándonos en la dependencia semántica o de contexto, para comprobar si Watson se comporta diferente vamos a utilizar textos de dos tipos, unos basados en opiniones sobre productos, donde esperamos que el significado de las palabras sea el común, y otros a los que llamaremos de *contexto competitivo*, formados por *tweets* generados durante campañas electorales donde el contexto, por ejemplo las noticias que ocurren durante la campaña, podrían afectar al significado.

Hay que recalcar que no buscamos necesariamente que Watson lo haga "peor" que el clasificador manual sencillo desarrollado por nosotros, al que llamaremos simplemente "Método Sencillo", sino que sería suficiente con que la diferencia entre ambos tipos de ejemplos varíe de forma notable.

En cuanto a la segunda pregunta, P2, tenemos que, si la respuesta a la primera pregunta es afirmativa y existen estos ejemplos de textos en los que Watson y el Método Sencillo obtienen diferencias notables, necesitaremos explicar a qué se debe esto. Para ello deberemos

analizar varios factores que pueden afectar a los resultados de los clasificadores, como los elementos que se utilizan con frecuencia en *Twitter* y que pueden aportar contexto a través de menciones, enlaces externos o los conocidos *hashtags*. Por otro lado, también se debe analizar si influyen otras características del texto como las palabras mal escritas que los sistemas pre-entrenados como Watson, al igual que pasa con la introducción de un idioma desconocido, pudieran no entender. Otro factor importante puede ser la complejidad del texto, pudiendo la eficiencia de los clasificadores depender de que textos sean más o menos difíciles de entender para un humano.

1.3. Análisis de sentimiento

A diferencia de las técnicas de recuperación de la información clásica, cuando hablamos de análisis de sentimiento no pretendemos obtener *hechos* sino *opiniones* (L^+), por lo que esta técnica se conoce también a menudo como "minería de opiniones" (*opinion mining*) (DLP03), aunque algunos autores distinguen ambos conceptos. Una definición informal pero suficiente para nuestros propósitos puede ser *el proceso que extrae pensamientos y percepciones humanos a partir de textos desestructurados* (HS19).

El conjunto de técnicas que permiten esta extracción de pensamientos y percepciones constituye una rama del aprendizaje automático, pero también de la minería de datos, del procesamiento de lenguaje natural, la lingüística computacional e incluso de la sociología y la psicología.

En particular, y ciñéndonos al aprendizaje automático, a menudo se han utilizado técnicas de aprendizaje supervisado para obtener el sentimiento de un conjunto de textos. El procedimiento habitual ha venido siendo:

1. Primero se selecciona (por lo general de forma aleatoria) un subconjunto de textos, el llamado *conjunto de entrenamiento*.

2. Un experto, generalmente un humano con conocimiento del tema, etiqueta estos textos con respecto a la opinión que expresan para un tema determinado. A menudo este etiquetado se limita a los valores negativos, representados por -1, neutros (0) o positivos (+1).

3. Los textos, junto con sus etiquetas, son empleados para generar un modelo utilizando alguna de las técnicas usuales de aprendizaje automático. Suele ser común que como paso previo los textos se *vectoricen* convirtiéndolos en vectores numéricos.

4. Finalmente el modelo se emplea para obtener el sentimiento del resto de los textos.

Este procedimiento general es largo y pesado, por lo que se han propuesto numerosas técnicas que utilizan modelos previamente entrenados. Aunque en este trabajo nos centraremos en el uso del servicio *IBM Watson* como método pre-entrenado para el análisis de sentimiento, existen otras muchas herramientas que ofrecen una funcionalidad similar, una de las más reconocidas y sencillas de utilizar es la biblioteca para *Python* llamada *TextBlob* (Lor) que ofrece múltiples funcionalidades de procesamiento de texto, entre ellas el análisis de sentimiento. Esta funcionalidad de la biblioteca asigna a un texto dado tanto una polaridad como una puntuación de subjetividad (AD17), siendo la polaridad un valor en el rango [-1, 1] y la puntación de subjetividad una valor en el rango [0,1], donde el 0 indica que es muy objetivo y el 1 que es muy subjetivo.

1.4. Estructura de la memoria

La estructura de la memoria se divide en cinco capítulos principales, donde el primero de ellos es esta introducción. En el capítulo siguiente se analizan las distintas metodologías de desarrollo de proyectos de data mining y se describe ampliamente la metodología utilizada, CRISP-DM, junto a su forma de aplicación en este trabajo. Seguidamente contamos con un capítulo dedicado al estudio de las métricas de evaluación más comunes en trabajos de este tipo, donde además se define una nueva métrica derivada de la métrica *Kappa de Cohen* que hemos denominado Kappa Penalizada y que se utilizará en el capítulo siguiente: el capítulo de experimentos, donde se muestran y se valoran los resultados intentando responder a las preguntas que se planteaban en la sección objetivos de este capítulo. Por último se tiene un capítulo con las conclusiones de este trabajo y el posible trabajo futuro.

Además de estos, esta memoria cuenta con dos capítulos adicionales que corresponden a las traducciones al idioma inglés de este capítulo de introducción y del capítulo de conclusiones y trabajo futuro.

El código *Python* desarrollado para la realización de este trabajo se encuentra en `https://github.com/raqblanc/analsis-sentimiento-contexto-competitivo/`.

Capítulo 2

Metodología de desarrollo utilizada

En este capítulo exponemos la metodología a seguir para el desarrollo del proyecto y la consecución de sus objetivos. En la primera sección comenzamos describiendo las metodologías más comunes en este tipo de trabajos, en la segunda presentamos la seleccionada para este trabajo, mientras que el resto de las secciones detallan cada una de las etapas que la conforman y su aplicación a nuestro problema concreto.

2.1. Metodologías más comunes

A día de hoy existen múltiples metodologías para la realización de proyectos de *ciencia de datos*, la mayor parte heredadas de las correspondientes en *data mining*, entre las más utilizadas (ver Figura 2.1) se encuentran *CRISP-DM*, *SEMMA* y *KDD Process*. Estas metodologías definen una serie de pasos secuenciales a seguir con la intención de guiar la implementación de aplicaciones de *data mining*.

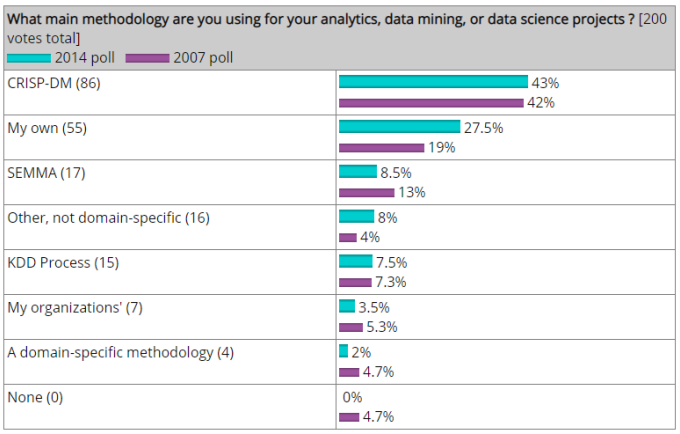

Figura 2.1: Ranking de metodologías más utilizadas (años 2007 y 2014)
Fuente: (Pia)

Antes de comenzar este trabajo examinamos las diferencias y similitudes entre estas tres metodologías. En cuanto a la metodología *SEMMA* y *KDD Process*, basándose en lo descrito en (AS08):

1. *KDD Process* (Figura 2.2) se divide en cinco etapas: una etapa de selección en la que se crea el conjunto de datos sobre el que se van a realizar los experimentos, una

etapa de preprocesamiento en la que se lleva a cabo la limpieza y la preparación del conjunto de datos creado en la etapa anterior, una etapa de transformación en la que se transforman los datos utilizando las técnicas necesarias, una etapa de minería de datos en la que se buscan los patrones o relaciones (modelos) presentes en los datos y finalmente una etapa de interpretación o evaluación en la que se analizan los modelos obtenidos en la etapa anterior. Este proceso es iterativo, una vez de completan todas las etapas y se tiene una evaluación, si esta es favorable se termina el proceso, de no ser así se volvería a retomar el proceso empezando por cualquiera de las etapas anteriores.

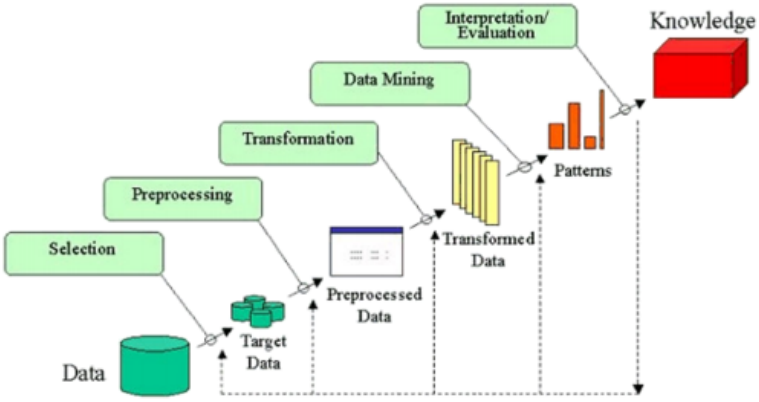

Figura 2.2: Etapas de la metodología *KDD Process*
Fuente: (LDS+ 13)

2. La metodología *SEMMA* (Figura 2.3) toma su nombre del acrónimo de sus cinco etapas: *Sample* o muestreo en la que se extrae una proporción de los datos suficiente para que esta contenga información significante, *Explore* o exploración en la que se exploran los datos buscando tendencias y anomalías con la intención de entender mejor los datos y obtener ideas, *Modify* o modificación en la que se modifican los datos de acuerdo con las variables seleccionadas para el proceso de modelización, *Model* o modelización en la que se crea el modelo dejando que el software busque automáticamente la combinación de datos que asegure unos buenos resultados de predicción y *Assess* o evaluación en la que se evalúan los datos realizando una valoración de cuán útiles y confiables son los modelos creados anteriormente. Al igual que ocurre en la metodología *KDD process*, puede ser necesario realizar algunas modificaciones, pudiendo regresar a etapas anteriores.

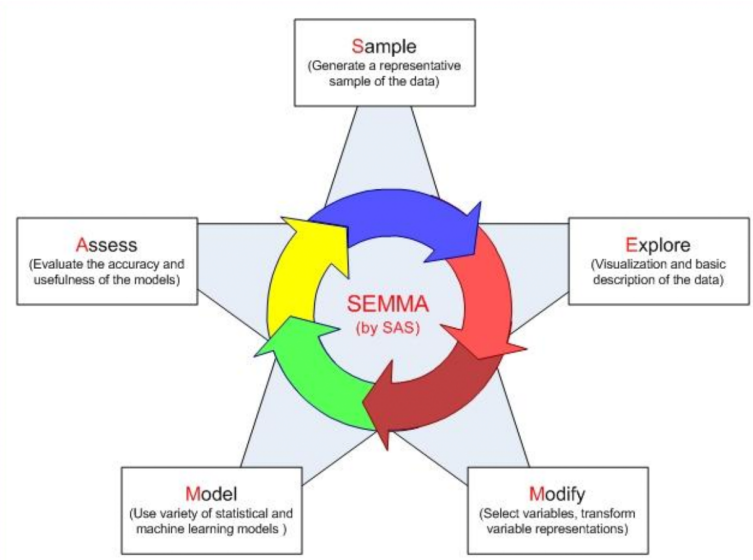

Figura 2.3: Etapas de la metodología *SEMMA*
Fuente: (TSD10)

Como se puede advertir, ambas metodologías son similares, incluso se puede observar una correspondencia directa entre las distintas etapas que las componen (ver Figura 2.4).

KDD	SEMMA	CRISP-DM
---	---	Business Understanding
Selection	Sample	Data Understanding
Preprocessing	Explore	
Transformation	Modify	Data Preparation
Data Mining	Model	Modeling
Interpretation/Evaluation	Assess	Evaluation
---	---	Deployment

Figura 2.4: Comparativa entre las etapas de las distintas metodologías descritas
Fuente: (Qua)

La última metodología por nombrar y primera en el ranking de la Figura 2.1 es *CRISP-DM*. La diferencia entre esta metodología y las dos anteriores es que estas se enfocan mucho más en las tareas de modelización dejando de lado los aspectos de negocio, mientras que *CRISP-DM* hace más hincapié en el entendimiento del negocio, e incluso cuenta con una etapa exclusivamente para ello, la etapa de *Business Understandig*, con la que no cuentan las otras dos metodologías, como puede observarse en la Figura 2.4.

El concepto de negocio que se trata en *CRISP-DM* sería equivalente a los objetivos que quieren probarse a través de los experimentos de este trabajo, con lo que la investigación y el entendimiento del "negocio" en este proyecto juega un papel importante. Es por esto y por ser una de las metodologías más utilizadas y más completas respecto al resto por lo que hemos decidido implementarla en este proyecto.

En lo que resta de capítulo se define a fondo esta metodología y sus etapas y se describe cómo ha sido implementada para nuestro caso particular: el análisis de sentimiento en textos.

2.2. Metodología CRISP-DM

CRISP-DM (WH00) del inglés *CRoss Industry Standard Process for Data Mining* se trata de una metodología de planificación para proyectos de minería de datos creada en el contexto de la *European Strategic Programme on Research in Information Technology (ESPRIT)*. Esta planificación se descompone en varias etapas secuenciales que pueden repetirse de forma cíclica (ver Figura 2.5):

1. Comprensión de negocio
2. Comprensión de los datos
3. Preprocesamiento
4. Modelización
5. Evaluación
6. Implantación

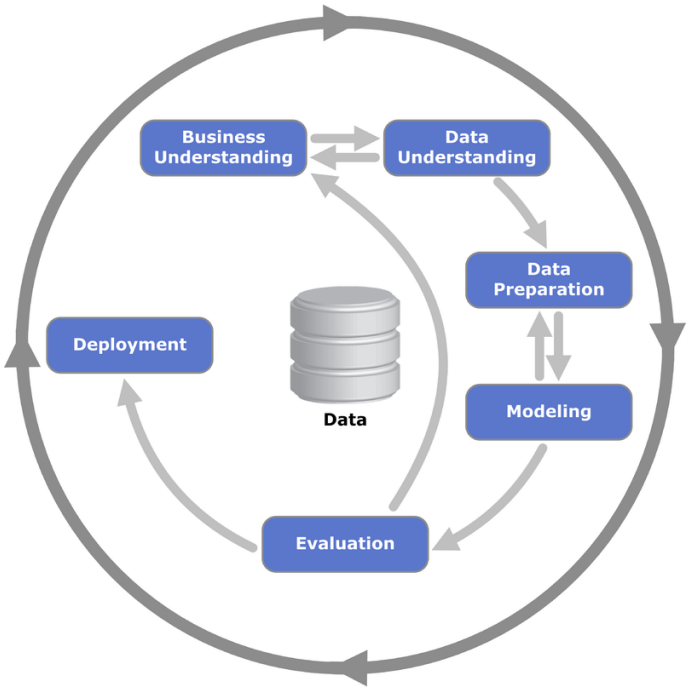

Figura 2.5: Etapas de la metodología CRISP-DM
Fuente: (Jen12)

Sin embargo, en la práctica es habitual que se lleven a cabo en un orden diferente, e incluso que se vuelva atrás retomando o sustituyendo una etapa que ya ha sido realizada.

En los siguientes apartados se van a describir brevemente cada una de las etapas de esta metodología. Esta descripción está basada principalmente en (Eur). Además, en cada caso explicaremos cómo se aplica esta etapa en nuestro proyecto concreto.

2.3. Comprensión del negocio

En esta primera etapa de *CRISP-DM* se intenta entender qué es lo que se quiere lograr desde la perspectiva del negocio, centrándose en los objetivos y los requisitos y comprendiendo qué factores pueden influir en el resultado final del proyecto. A partir de esta comprensión del negocio, se lleva a cabo la definición del problema de minería de datos y se diseña el plan de actuación para abordar el problema. Se evalúa la situación actual, listando los recursos disponibles (datos, software, hardware...), los requisitos, riesgos, un glosario de términos, etc.

Además, en esta etapa también se debe definir el criterio de éxito, el que nos dirá si del proyecto ha finalizado exitosamente o no.

En este trabajo el objetivo ha sido ser capaz de determinar si el acierto en análisis de sentimiento mediante herramientas previamente entrenadas como Watson (ver apartado 2.6.1) depende del contexto en el que se generan las opiniones a examinar. Para ello se ha decidido comparar Watson con un analizador de sentimiento muy básico pero que utiliza etiquetas generadas manualmente en cada caso. Como comentamos en la introducción, para comparar ambos métodos utilizaremos conjuntos de datos de dos tipos:

1. Comentarios generados durante eventos competitivos tales como elecciones políticas, concursos, etc. En particular en nuestro caso emplearemos *tweets* generados durante procesos electorales. La idea es que estos conjuntos tienen una fuerte dependencia del contexto; son eventos que suceden en un periodo de tiempo corto y donde los usuarios expresan opiniones acerca de lo que ha ocurrido en la campaña cuya interpretación puede ser compleja para una herramienta automática. Además al ser eventos competitivos los mensajes a menudo aluden a varios contrincantes lo que puede confundir fácilmente a Watson, sobre todo si el contrincante no se nombra explícitamente.

2. Reseñas de productos o marcas generadas por usuarios. En este tipo de datasets el contexto influye menos al no haber tanta interacción entre usuarios ni referirse éstos a sucesos que aparezcan, por ejemplo, en medios de comunicación. Además el sujeto del mensaje suele ser único lo que elimina la posible confusión y en estas reseñas los usuarios valoran un producto de forma objetiva, centrándose en él, lo que facilita la labor de las herramientas automáticas.

El objetivo de este trabajo y lo que se espera con la realización de estos experimentos es que estos dos conjuntos de datos de naturaleza diferente permitan mostrar alguna diferencia en la comparativa entre Watson y el Método Sencillo basado en clasificación manual.

Como criterio de éxito para la evaluación del cumplimiento de los objetivos se ha realizado un estudio de las métricas más comunes para este tipo de trabajos que se describe con profundidad en el capítulo 3.

2.4. Comprensión de datos

En esta fase se adquieren los datos que fueron listados en la etapa anterior como recursos del proyecto. Además, se incluyen todas las acciones que facilitan la comprensión del dato, como el almacenamiento o la carga en algún sistema.

Algunas de las actividades de comprensión del dato pueden ser:

- Listado de los distintos orígenes de datos, sus localizaciones y el método utilizado para su adquisición.

- Descripción de los datos (formato, cantidad, tipo de variables...)
- Exploración profunda de los datos, realizando preguntas sobre minería de datos a través de consultas o visualizaciones.
- Verificación de la calidad de los datos, realizando preguntas que cuestionen su calidad teniendo en cuenta los requisitos, si contienen errores o si existen *missing values*.

2.4.1. Tipos de variables

Como acabamos de comentar, a la hora de estudiar los datos es importante conocer sus características antes de analizarlos. Para ello, resulta fundamental determinar cuál es el tipo de cada variable con la que vamos a tratar de entre los cuatro que se suelen considerar al realizar experimentos, y que enumeramos a continuación siguiendo la descripción dada en (FH02):

- Nominales o categóricas: son las variables que, cuando dos objetos son equivalentes en algún aspecto les asigna el mismo nombre o número. De asignársele un número, se debe tener en cuenta que no existe relación entre la magnitud del número y lo que se está midiendo, solo puede decirse si otro objeto tiene el mismo número o no lo tiene. Por tanto, estos números son etiquetas y no deberían emplearse para realizar operaciones, porque no tienen un significado numérico y la única operación que tenemos predefinida sobre este conjunto es la igualdad.

  Por ejemplo: supongamos que entre los miembros de un equipo de fútbol estamos midiendo cuál es mejor jugador. Tenemos al jugador número 7, que es centrocampista, y al jugador número 1, que es delantero; pero esto no significa que el jugador número 7 sea mejor que el jugador número 1 por tener un número mayor.

  La única utilidad de las etiquetas, en este caso numéricas, sería la de observar las frecuencias. Por ejemplo, en el supuesto del equipo de fútbol, saber cuántos goles ha marcado un jugador con respecto a otro. Obsérvese que a estos efectos igual que se utilizan números se podrían utilizar nombres o letras.

- Ordinales: estas variables proporcionan más información que las variables categóricas. Si utilizamos una escala ordinal para medir algo podremos decir no solo las cosas que han ocurrido si no también en qué orden han ocurrido. Sin embargo, estas variables no nos aportan información acerca de las diferencias entre los valores.

  Por ejemplo, supongamos una encuesta de satisfacción a los asistentes de una conferencia. A la pregunta: "¿cómo de útil le ha resultado esta conferencia?" tendríamos como respuesta la variable ordinal que puede tomar uno de estos valores: "muy poco", "poco", "regular", "mucho", "muchísimo". Estas etiquetas nos aportan información en una escala ordinal, el orden nos indica que si un asistente elige "mucho", esto es mejor que si ha elegido "poco". Sin embargo, no podemos afirmar que haya la misma distancia entre distintas etiquetas, por ejemplo, entre elegir "muy poco" y "poco" y entre elegir "regular" o "mucho" no podemos decir que exista la misma distancia, como tampoco podemos decir que a un asistente que ha marcado "mucho" le ha parecido el doble de útil que a uno que ha marcado "poco".

  Un tipo particular de variable ordinal son los resultados de encuestas que utilizan la llamada *escala de Likert* (Wue05).

- Intervalo: son unos de los tipos de variables más utilizados, ya que dan más información que las variables ordinales. En una escala de intervalos, la distancia entre intervalos es equivalente, es decir, la distancia es la misma entre cada opción. Sin embargo, con estas variables no se puede hablar de, por ejemplo: "el doble", ya que estas escalas no tienen un valor absoluto cero, sino que pueden tener valores negativos.

  Esto ocurre, por ejemplo, en el caso de la temperatura. Si utilizamos la escala Celsius para medir, podremos estar seguros de que la diferencia entre 20° y 25° es la misma que entre 43° y 48° (son 5° de diferencia en ambos y esos 5° de diferencia son equivalentes). Sin embargo, no podríamos decir que estar a 20° grados es -2 veces estar a -10°, igual que no podríamos decir que estar a 40° es que haga el doble de calor que a 20°.

- Ratio: estas variables son similares a las de intervalo, con la diferencia de que en este caso sí se tiene un valor absoluto cero, por lo que no existen valores negativos y se puede utilizar la proporcionalidad entre valores.

  Un ejemplo sencillo sería la escala de los centímetros. Supongamos que tenemos la longitud del ancho de unas mesas. Podremos decir, en primer lugar, que una mesa de 35 cm y una de 45 cm tienen la misma diferencia de longitud que una mesa de 50 cm y una de 60 cm (5 cm), pero también podremos decir que una mesa de 40 cm mide el doble que una de 20 cm y un tercio que una de 120 cm, cálculo que no podía realizarse en la escala de las variables de intervalo.

En este trabajo nos proponemos asignar un "sentimiento" a cada mensaje. Este sentimiento vendrá representado por etiquetas del estilo -1 para negativo, 0 para neutro y +1 para positivo. Se trata entonces de una variable de tipo ordinal; un sentimiento +1 es más positivo que un sentimiento 0, y este último más que un sentimiento -1. En principio no serían variables de tipo intervalo, porque no podemos decir que la distancia entre -1 y 0 sea la misma que la que hay entre 0 y 1. Sin embargo, a efectos prácticos, nos interesará contar la "distancia" en el sentido de número de valores intermedios, como una forma de cuantificar el término "más lejos". Así, diremos que hay una distancia de 2 entre -1 y +1, ya que hay dos "saltos": el de -1 a 0, y el de 0 a +1. En cambio la distancia entre -1 y 0, o entre 0 y 1 es de 1, porque son valores consecutivos en el orden. Es fácil comprobar que dado un orden total esta definición corresponde en efecto a una distancia en el sentido matemático. Esta distancia, inducida por la relación de orden total, será importante a la hora de elegir una métrica que penalice más los errores más "lejanos". Hay que observar sin embargo que se trata de una decisión arbitraria y que podrían elegirse otras, como por ejemplo el cuadrado del número de valores intermedios.

2.4.2. Conjuntos de datos utilizados

Los conjuntos de datos de partida disponibles en este trabajo han sido cuatro, dos de cada tipo de los señalados anteriormente: eventos competitivos y reseñas de productos.

Del primer tipo consideramos *tweets* de dos campañas electorales:

1. Descripción: *tweets* etiquetados manualmente sobre las elecciones de Estados Unidos de 2020 que hacen referencia a alguno de los dos candidatos a presidente: Joe Biden o Donald Trump, le denominaremos *elecc-2020*. Sus características son:

   - Formato: Excel (*.xlsx*)

- Tamaño: 2088 registros
- Columnas de interés:
  - Texto: texto del *tweet* útil para el análisis de sentimiento.
  - Opinión Biden: etiqueta manual valorando la opinión sobre Joe Biden que refleja el *tweet*, pudiendo ser esta -1 (negativa), 0 (neutro) o 1 (positiva).
  - Opinión Trump: etiqueta manual de la opinión sobre Donald Trump que refleja el *tweet*, pudiendo ser esta -1 (negativa), 0 (neutro) o 1 (positiva).

2. Descripción: *tweets* etiquetados manualmente sobre las elecciones de Estados Unidos de 2016. Se parte de tres ficheros de origen: *elecc-2016-zero* para los etiquetados como neutrales, *elecc-2016-pos* para los positivos y *elecc-2016-neg* para los negativos hacia Donald Trump. A este conjunto le denominaremos *elecc-2016*. Sus características son:

   - Formato: texto (*txt*)
   - Tamaño: positivos 807, negativos 445 y neutros 1728.
   - Columnas de interés:
     - Texto: texto del *tweet* útil para el análisis de sentimiento.
     - Opinión Trump: etiqueta manual de la opinión sobre Donald Trump que refleja el *tweet*, pudiendo ser esta -1 (negativa), 0 (neutro) o 1 (positiva).

En cuanto al segundo tipo, reseñas de productos, consideramos también dos conjuntos de datos.

1. Descripción: Reseñas de Amazon sobre aplicaciones Android obtenidas de (McA) etiquetadas por los propios usuarios que escribieron la reseña. A este conjunto le denominaremos *reseñas-amazon*. Sus características son:

   - Formato: *json*
   - Tamaño: 752937 registros
   - Columnas de interés para nuestro proyecto:
     - *reviewText*: texto útil para el análisis de sentimiento
     - *overall*: opinión del usuario del 1 al 5, significando el 1 que la opinión es lo más negativa posible y el 5 la más positiva.

2. Descripción: *tweets* con opinión de usuarios sobre distintas aerolíneas obtenidos de (Jha). Sus características son:

   - Formato: *csv*
   - Tamaño: 14872 registros
   - Columnas de interés:
     - *text*: texto del *tweet* útil para el análisis de sentimiento.
     - *airline sentiment*: opinión del usuario con valores "positive" (positiva), "negative" (negativa) y "neutral" (neutro).

Es importante notar que entre el segundo grupo de ejemplos hay uno que corresponde a *tweets*, como ocurre con los dos ejemplos del primer grupo. De otra forma podríamos tener dudas, en caso de encontrar diferencias entre el comportamiento de Watson y nuestro clasificador, sobre si estas diferencias pudieran deberse a las características particulares de *Twitter*.

2.5. Preprocesamiento

En esta etapa se cubren todas las acciones que deben realizarse para obtener el conjunto de datos final con el que se van a realizar los experimentos. Algunas de estas acciones pueden ser:

- Selección de los datos que van a formar parte del conjunto final. El criterio para seleccionar o no los datos debe ser acorde con los objetivos, la calidad y las restricciones técnicas (volumen de datos, computación, tipo de datos...). Además, se realiza la selección de los atributos (columnas) y de los registros (filas) de cada tabla.
- Limpieza de los datos para elevar su nivel de calidad.
- Construcción de los conjuntos de datos finales, realizando las pertinentes transformaciones, alteraciones, combinaciones o agregaciones de columnas...
- Integración de toda la colección de datos utilizando métodos para la combinación de datos originarios de múltiples fuentes.

En este trabajo se han seleccionado los conjuntos del apartado 2.4.2, que han sufrido las siguientes transformaciones:

- De los dos conjuntos del primer tipo, de campañas electorales:
  1. Del conjunto *elecc-2020* se han seleccionado únicamente las columnas: texto, opinión Biden y opinión Trump. A partir de estas tres columnas se han generado dos ficheros en formato *csv*, uno que contiene todos los *tweets* y los datos de la columna opinión Trump y otro que contiene todos los *tweets* y los datos de la columna opinión Biden. A estos nuevos conjuntos de datos les denominaremos *trump-2020* y *biden-2020*.
  2. Del conjunto *elecc-2016*, utilizando los tres ficheros de texto (tweets positivos, negativos y neutros), se ha creado un fichero *csv* de dos columnas: la primera con los textos contenidos en cada uno de los conjuntos anteriores y otra con valores 0, 1 o -1 dependiendo del fichero de origen *elecc-2016-zero*, *elecc-2016-pos* o *elecc-2016-neg* respectivamente, que corresponden con las opiniones hacia Trump. A este nuevo conjunto de datos le denominaremos *trump-2016*.
- De los conjuntos de eventos competitivos:
  1. Seleccionamos únicamente los 4000 primeros registros del conjunto *reseñas-amazon* y las columnas *reviewText* y *overall*, que corresponden al texto y a la opinión. A partir de los datos de estas dos columnas generamos un nuevo fichero *csv*. A este conjunto le denominaremos *amazon-4000*.
  2. Seleccionamos únicamente los registros (filas) cuya columna *airline* coincide con "Delta" o "American". Después, se transforma el valor de la columna "airline sentiment": se sustituye "positive", "negative", "neutral" por los valores 1, -1 y 0 respectivamente. A partir de estos datos, se crean dos ficheros *csv*, cada uno con la información de una sola aerolínea. A estos conjuntos de datos les denominaremos *aerolinea-delta* y *aerolinea-american*.

Por lo tanto, tendríamos un total de seis ficheros *csv* que conformarían los seis conjuntos de datos con los que se van a realizar los experimentos: *trump-2020*, *biden-2020*,

trump-2016, *amazon-4000*, *aerolinea-delta* y *aerolinea-american*. Todos los ficheros tienen la misma estructura de dos columnas: texto (el texto a analizar, en inglés) y sentimiento (valores en {-1, 0, 1} para todos los conjuntos excepto para *amazon-4000* que los valores están en {1, 2, 3, 4, 5}).

Todos estos conjuntos de datos ya preprocesados y limpios pueden encontrarse en `https://github.com/raqblanc/analsis-sentimiento-contexto-competitivo/tree/main/data`.

2.6. Modelización

En esta etapa se realizan las siguientes acciones:

1. Selección de las técnicas de modelización que van a ser utilizadas para realizar los experimentos. Es decir, elección de qué mecanismos se van a utilizar para obtener los modelos de aprendizaje automático.

2. Generación de un procedimiento o mecanismo mediante el cuál se pueda obtener la calidad y validez de las pruebas realizadas. Por ejemplo, para métodos de aprendizaje supervisado un procedimiento común es la creación de un conjunto de prueba o *test* obtenido a partir del conjunto de datos total y un conjunto de entrenamiento o *test*. Además, en esta etapa debe definirse un criterio de evaluación que nos permita obtener la calidad y validez de los experimentos realizados.

3. Creación de los modelos a través de alguna herramienta, teniendo en cuenta y anotando el ajuste de parámetros para su posterior evaluación.

4. Evaluación de los modelos utilizando el criterio de calidad definido, pudiendo comparar los distintos métodos y los distintos ajustes de parámetros. Se debe iterar entre los puntos 3 y 4 hasta haber encontrado con seguridad modelos cuyo resultado es bueno.

En este trabajo se han utilizado únicamente dos técnicas de modelización, las dos que se quieren comparar: el servicio de *IBM Watson para análisis de sentimiento* y un método sencillo de elaboración propia para análisis de sentimiento basado en la técnica de *machine learning Random Forest*. Ambas técnicas han sido desarrolladas en código *Python* se detallan en los dos siguientes subapartados.

Tanto la metodología de desarrollo de estas técnicas como los experimentos de análisis de sentimiento para los conjuntos descritos en el subapartado 2.5 se describen con detalle en el capítulo 4.

2.6.1. IBM Watson

En los últimos años se está experimentando en el mundo de la informática una avalancha de información no estructurada. Para abordar el manejo y el análisis de todos estos datos se están implementando nuevos sistemas: los sistemas cognitivos o *cognitive systems* (Hig12). Uno de los sistemas más reconocidos en este ámbito es *IBM Watson* (que por simplicidad en adelante llamaremos Watson), ya que su precisión en el análisis de texto es similar a la humana e incluso su velocidad de respuesta es mayor a la de los seres humanos. Sin embargo, Watson no conoce individualmente cada palabra de un idioma, si no que comprende las características del lenguaje que utilizamos las personas para comunicarnos. Como ejemplo

de la potencia de este sistema, en 2011 Watson participó en el famoso concurso estadounidense de preguntas y respuestas *Jeopardy!* (Fer12), en el cual tres concursantes responden a preguntas en lenguaje natural que requieren una rápida comprensión, consiguiendo batir a los dos mejores jugadores hasta el momento. Otro ejemplo es el debate llevado a cabo por IBM (SBA+21) donde Watson, al que en este proyecto llamaron *Project Debater*, compitió contra el experto en debates H. Natarajan acerca de la subvención de la educación preescolar.

Watson no se trata solo de un sistema interno utilizado por IBM, sino que existe como herramienta para usuarios y empresas. Está comprendido por una serie de servicios en la nube (*IBM Cloud*). Para este trabajo se ha utilizado en concreto el servicio *Natural Language Understanding-3s*, al que se ha accedido a través de una API con *Python*. Las llamadas a esta API contienen como entrada el texto a analizar (correspondiente a los datos contenidos en cada uno de los conjuntos definidos en el apartado 2.4.2), el idioma (inglés) y el *target* u objetivo del que se quiere obtener el sentimiento; este último es opcional, se permite obtener el sentimiento hacia un *target* concreto o de forma general en el texto. De la respuesta obtenida de la API se ha tenido en cuenta el *score* o puntuación ($-1 \leq score \leq 1$, donde -1 es el sentimiento más negativo y 1 el más positivo) y el *sentiment* o sentimiento, que representa el sentimiento en lenguaje natural, pudiendo ser *negative*, *neutral* o *positive* dependiendo del rango de *score* en el que se encuentre el resultado.

2.6.2. Método Sencillo

Para comprobar que para el tipo de textos de contexto competitivo Watson no proporciona tan buenos resultados en el análisis de sentimiento como sí lo hace en contextos más sencillos como el de reseñas de productos se ha utilizado un algoritmo sencillo, en este caso *Random Forest* (Ho95), aunque podría haberse elegido cualquier otro algoritmo de *machine learning* .

Es importante señalar que este método no está optimizado para obtener el mejor rendimiento posible, ya que este no es el objetivo del trabajo. Lo se quiere es comparar cómo cambia el acierto entre los datasets de los dos tipos seleccionados para un clasificador que entrena a partir de etiquetas generadas para ese caso concreto, como este, o para una herramienta previamente entrenada como Watson.

2.7. Evaluación

En esta penúltima etapa se realiza la evaluación del modelo o de los modelos elegidos con respecto a los objetivos de negocio. Antes de realizar la implementación final del modelo, es importante evaluar minuciosamente el modelo comprobando los pasos seguidos para su construcción y entendiendo en qué medida el o los modelos elegidos abordan todos y cada uno de los objetivos de negocio. Esto nos permitirá tomar una decisión sobre el uso que se va a hacer de los resultados finales.

En nuestro caso haremos esta valoración de resultados en el capítulo 4 de experimentos.

2.8. Implantación

Una vez tomada una decisión, se toman los resultados obtenidos de la evaluación y a partir de ellos se realiza un informe que resume todos los resultados y presenta las conclusiones obtenidas. De tratarse de un proyecto enfocado a que clientes consuman estos

resultados, se determinaría una estrategia a seguir para su implantación y su futuro mantenimiento, creando un plan de implantación y un plan de mantenimiento. Por último se realiza una revisión del proyecto, comprendiendo lo que ha ido bien y lo que ha ido mal a lo largo del proyecto y cómo podría mejorarse.

En este trabajo, los informes requeridos en esta etapa equivaldrían a lo narrado en el capítulo 5 de conclusiones y el trabajo futuro.

Capítulo 3

Métrica de evaluación

En este capítulo revisamos posibles métricas para el problema que nos ocupa y definimos nuestra propia métrica, variante de la conocida *Kappa*, a la que llamaremos *Kappa Penalizada*, denotada por $p\kappa$.

3.1. Métricas más comunes

3.1.1. La importancia de las métricas

Dentro de la primera etapa de la metodología *CRISP-DM*, la *comprensión de negocio*, la definición de una métrica adecuada constituye un punto muy importante por dos razones principales:

- Permite comparar unos modelos con otros para establecer cuál es el mejor según un criterio predeterminado.
- Permite ver si el modelo obtenido finalmente alcanza una meta predeterminada; por ejemplo, podemos desear que una alarma de incendios detecte el 95 % de las emisiones de cierto gas cuando pasa de una cota dada, y que el número de falsos positivos (detecciones incorrectas por debajo de la cota) sea menor al 0.05 %.

En nuestro caso nos vamos a concentrar sobre todo en el primer punto; no tenemos una meta inicial pero sí deseamos comparar la bondad de diferentes modelos.

3.1.2. Matriz de confusión

Dos de las métricas más comunes para evaluar modelos de clasificación supervisados son las métricas *Precision* y *Recall*, que analizaremos a continuación. Estas y otras métricas que veremos después parten de la llamada *matriz de confusión*.

La matriz de confusión, que a partir de ahora denotaremos como F, es útil para visualizar y comprender lo bien que clasifica un modelo un conjunto de valores de tamaño T (normalmente el *conjunto de test*) previamente etiquetados. Denotaremos por y_{real} el conjunto de valores que indican la etiqueta real, que asumimos como correcta, y que a menudo se ha generado manualmente o por algún medio externo fiable. Denotaremos por y_{pred} al conjunto de valores que indican la etiqueta que ha asignado el modelo que queremos evaluar. La matriz de confusión es entonces una matriz de dimensiones $C \times C$ con el siguiente aspecto:

		Predicción			
		$etiqueta_1$	$etiqueta_2$	...	$etiqueta_C$
Realidad	$etiqueta_1$				
	$etiqueta_2$				
	...			$v_{i,j}$	
	$etiqueta_C$				

donde C es el número de etiquetas diferentes o clases. El valor $v_{i,j}$ de una celda, donde i $(1 \leq i \leq C)$ es la fila y j $(1 \leq j \leq C)$ es la columna, indica cuántos de los valores se han clasificado por el modelo como $etiqueta_j$ cuando su etiqueta real es la $etiqueta_i$.

Por convenio, haremos que las filas correspondan a los valores reales y las columnas a los predichos, aunque en algunos trabajos otros autores pueden usar otro convenio.

El uso de esta matriz es muy interesante ya que permite la visualización e interpretación del modelo de una manera sencilla y rápida. Además, podemos deducir a partir de ella información interesante, como el total de aciertos del modelo (sumando los valores de la diagonal), el número de valores reales para cada etiqueta (sumando los valores de cada fila) o el número de valores predichos para cada etiqueta (sumando los valores de cada columna).

Para una mayor comprensión de este concepto, vamos a introducir un ejemplo de clasificador de sentimiento expresado en texto. Las etiquetas en este caso serían: sentimiento positivo (1), negativo (-1) o neutro (0), con que $C = 3$. Supongamos que partimos de un conjunto de test de tamaño $T = 6$ previamente etiquetado[1]:

$$y_{real} = \{-1, -1, -1, 0, 1, 1\}$$

y que el modelo de clasificación generado que queremos evaluar nos devuelve el conjunto de predicciones:

$$y_{pred} = \{-1, 0, 0, 0, 1, -1\}$$

La matriz de confusión resultante al comparar ambos conjuntos sería la que podemos observar en la Tabla 3.1 en amarillo. A partir de esta, podemos observar, entre otras cosas, que el modelo ha acertado la clasificación de 3 textos, que ha predicho como neutro dos textos que eran negativos, que el total de textos que ha predicho como neutros es 3, etc. Además, es muy útil calcular la cantidad de elementos con etiqueta real de cada tipo (en naranja) y la cantidad de elementos con etiqueta predicha de cada tipo (en rosa).

		Predicción			Suma real
		-1	0	1	
Realidad	-1	1	2	0	3
	0	0	1	0	1
	1	1	0	1	2
Suma predicción		2	3	1	6

Tabla 3.1: F para el ejemplo de análisis de sentimiento

Otra matriz útil para los cálculos de medidas es la llamada *matriz de confusión normalizada*, o simplemente matriz normalizada, que denotaremos como N, se calcula normalizando los valores de la matriz de confusión F, es decir, dividiendo cada uno de sus valores por el número total de elementos T.

[1] Asumimos que los conjuntos están ordenados, es decir que la etiqueta predicha que aparece en primer lugar corresponde al primer valor real, y así sucesivamente.

Definición 3.1.1 *Sea F una matriz de confusión. Su matriz normalizada asociada N se define a partir de sus elementos como.*

$$N_{i,j} = \frac{F_{i,j}}{sum(F)}$$

$$donde\ sum(F) = \sum_{i=1}^{C}\sum_{j=1}^{C} F_{i,j}$$

Utilizaremos la abreviatura $sum(X)$ para otras matrices X, siempre con el mismo significado de suma de todos sus elementos.

En nuestro ejemplo de análisis de sentimiento, la matriz N sería la que puede observarse en la Tabla 3.2.

		Predicción			Suma real
		-1	0	1	
Realidad	-1	$0{,}1\widehat{6}$	$0.\widehat{3}$	0	0.5
	0	0	$0{,}1\widehat{6}$	0	$0{,}1\widehat{6}$
	1	$0{,}1\widehat{6}$	0	$0{,}1\widehat{6}$	$0.\widehat{3}$
Suma predicción		$0.\widehat{3}$	0.5	$0{,}1\widehat{6}$	1

Tabla 3.2: N para el ejemplo de análisis de sentimiento

3.1.3. Precision

La métrica *Precision* (precisión) indica la proporción de veces que un modelo de clasificación acierta al predecir que un elemento tiene como etiqueta x. Su cálculo se realiza como el número de aciertos (valor de la celda $F_{x,x}$) entre el número total de elementos que ha predicho con etiqueta x (suma de la columna con $etiqueta_x$), que utilizando la matriz de confusión F, equivaldría al siguiente cálculo:

$$precision(x) = \frac{F_{x,x}}{F_{\bullet x}}$$

donde la notación $F_{\bullet x}$ representa la suma de la columna x, esto es:

$$F_{\bullet x} = \sum_{i=1}^{C} F_{i,x}$$

A partir de ahora usaremos esta notación y su análoga $F_{x\bullet}$ para la suma de filas cuando sea conveniente.

Volviendo a nuestro ejemplo anterior, y teniendo como referencia la Tabla 3.1, podemos calcular la precisión con la fórmula anterior, obteniendo: $precision(-1) = 1/2$, $precision(0) = 1/3$ y $precision(1) = 1$.

3.1.4. Recall

La métrica *Recall* (exhaustividad) indica la proporción de los datos con etiqueta real x que han sido clasificados por el modelo correctamente. Para el cálculo de esta métrica, al igual que hacíamos para *Precision*, utilizamos la matriz de confusión F. En este caso, la fórmula sería:

$$recall(x) = \frac{F_{x,x}}{F_{x\bullet}}$$

Es decir, el número de aciertos (valor de $F_{x,x}$) entre el total de elementos etiquetados realmente con etiqueta x (suma de los valores de la fila con $etiqueta_x$).

Para el ejemplo de análisis de sentimiento anterior, utilizando la F de la Tabla 3.1 y la fórmula, obtendríamos $recall(-1) = 1/3$, $recall(0) = 1$ y $recall(1) = 1/2$.

3.1.5. F-score

Aunque *Recall* y *Precision* son dos de las métricas más comunes, ambas poseen una gran desventaja, y es que por sí solas no dan una información completa; a menudo se puede obtener un excelente *Recall* a costa de penosa *Precision* para una etiqueta dada. Por ejemplo, en nuestro caso, podríamos hacer que todos los mensajes se clasificaran como negativos (-1); en este caso el *Recall* para la etiqueta -1 sería 1, porque en efecto todos los mensajes realmente negativos se habrán clasificado como tales, pero la precisión para esta misma etiqueta bajará mucho. Es por eso que se suelen buscar métricas únicas y con significado por sí mismas como son *F-score* y *Kappa*, que veremos con detalle a continuación.

El *F-score* (*valor-F*) es una medida calculada como la media armónica de *Precision* y *Recall*. Denotaremos al *F-score* como F_β, donde β determina la importancia (ponderación) que se le da al *Recall* y a la *Precision*; si $\beta > 1$ se le da más importancia al *Recall*, si $\beta < 1$ al *Precision* y si $\beta = 1$ se da la misma ponderación a ambas. Su fórmula es la siguiente:

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{\beta^2 \cdot precision + recall}$$

Sin embargo, esta medida sigue siendo aplicable a cada clase (Pow15), y no una medida única para todo el experimento. Además, un problema añadido es que para conjuntos con mucho sesgo o muy desequilibrados entre clases, nos puede salir un F alto en la clase con más valores. Por ejemplo imaginemos un conjunto con 100 valores, 99 etiquetados como A y 1 como B. Entonces un clasificador que marque cualquier valor como A tendrá una alta precisión ($\frac{99}{100}$) y un *Recall* de 1, por lo que saldrá un F_β muy alto. Esto no significa sin embargo que el clasificador esté funcionando muy bien para esa clase. Por ello debemos buscar una media que tenga en cuenta estos desequilibrios.

Para una comparativa detallada entre distintas métricas ver (Pow20).

3.2. Kappa

Definido por primer vez en (Coh60), el coeficiente Kappa de Cohen se define como una medida estadística que tiene en cuenta el factor aleatorio, lo que ocurre por azar. De esta forma se consideran los casos con valores desequilibrados entre clases, donde por azar es más probable elegir la clase con más etiquetas. Además, a diferencia de las métricas vistas hasta ahora (con la excepción de *F-score*) se trata de un valor único que no depende de la clase concreta.

Se utiliza cuando los elementos a predecir son cualitativos para comparar cuánto coinciden dos clasificadores. Esto se realiza comparando la coincidencia observada en la matriz de confusión con una que se obtendría al azar.

Esta métrica, que denotaremos κ, busca determinar coincidencias que no se deben al azar entre dos clasificadores, por ello se habla de dos valores: la coincidencia observada, que denotaremos *co* y la coincidencia por casualidad, que denotaremos *cc*.

En *machine learning* es habitual que en lugar de utilizar κ para comparar dos modelos, se utilice para comparar un solo modelo con el "juez perfecto", el que predeciría todo correctamente. La idea es que cuanto más nos aproximemos a ese clasificador ideal, mejor

será nuestro modelo, y que κ nos muestra ese grado de aproximación. Su interpretación usual es la siguiente: un valor de κ entre 0 y 0.20 se considera poco, entre 0.21 y 0.40 bastante, entre 0.41 y 0.60 moderado, entre 0.61 y 0.80 sustancial y entre 0.81 y 1 perfecto. Un valor negativo indica que el modelo acierta menos de lo que esperaríamos por azar, lo que seguramente nos indica que hay algún error (etiquetas intercambiadas, etc).

La coincidencia observada *co* es la coincidencia de los dos clasificadores en la práctica, la que puede observarse en la matriz de confusión, mientras que la coincidencia por casualidad es la probabilidad de acierto por casualidad; ambas con valores entre 0 y 1. Podemos definir κ a partir de estos dos valores como sigue.

Definición 3.2.1 *Dada una coincidencia observada co y una coincicidencia por casualidad cc la Kappa de Cohen se define como:*

$$\kappa = \frac{co - cc}{1 - cc}$$

Podemos entender $(1 - cc)$ como el camino que queda por recorrer, entre el azar y la coincidencia perfecta. Por ejemplo, si tenemos que $cc = 0{,}2$, si creamos un clasificador aleatorio, este acertará solo un 20 % de las veces, es decir, el 80 % que se falla por el azar, es lo que podemos mejorar. Si además tuviéramos que $co = 0{,}4$, significaría que nuestro clasificador acierta un 40 % de las veces. Así, tendríamos $\kappa = \frac{0{,}4-0{,}2}{1-0{,}2} = 0{,}25$, significando esto que el modelo lo hará un 25 % mejor que el azar.

Para el cálculo de *co* y *cc* es necesaria la matriz de confusión F, definida en el apartado anterior, y la matriz de confusión normalizada N.

Una vez calculada N, *co* se traduciría como la suma de los valores de la diagonal, ya que se trata de los aciertos del clasificador.

Definición 3.2.2 *Dada una matriz de confusión normalizada N, la coincidencia observada co se define como:*

$$co = \sum_{i}^{C} N_{i,i}$$

Mientras que *cc* se calcula como sigue.

Definición 3.2.3 *Dada una matriz de confusión normalizada N, la coincidencia por casualidad cc se define como:*

$$cc = \sum_{i=1}^{C} (N_{\bullet i} \cdot N_{i\bullet})$$

Para un mayor entendimiento, vamos a calcular κ en el ejemplo de análisis de sentimiento en texto que hemos utilizado anteriormente. En esta ocasión, partiendo de la F de la Tabla 3.1, calculamos la matriz normalizada N (ver Tabla 3.2).

Así, para nuestro ejemplo tendríamos:

$$co = 0{,}1\widehat{6} + 0{,}1\widehat{6} + 0{,}1\widehat{6} = 0{,}5$$

$$cc = 0.\widehat{3} \cdot 0.\widehat{5} + 0.\widehat{5} \cdot 0{,}1\widehat{6} + 0{,}1\widehat{6} \cdot 0.\widehat{3} = 0{,}30\widehat{5}$$

por lo que

$$\kappa = \frac{0{,}5 - 0{,}30\widehat{5}}{1 - 0{,}30\widehat{5}} = 0{,}28$$

Existe otra definición equivalente para el cálculo de κ utilizando N y una nueva matriz generada a partir de la anterior: la matriz de probabilidades marginales M.

Definición 3.2.4 *Dada una matriz normalizada N, la matriz de probabilidades marginales M se define como:*

$$M_{i,j} = N_{i\bullet} \cdot N_{\bullet j}$$

Es inmediato comprobar que esta nueva matriz nos permite reescribir *cc* como sigue.

Proposición 3.2.1 *Dada una matriz de probabilidades marginales M, cc se puede definir como:*

$$cc = \sum_{i=1}^{C} M_{i,i}$$

y esta definición es equivalente a la definición 3.2.3.

La demostración es directa comparando ambas definiciones y la de M.

Además, la matriz M también nos permite reescribir κ como sigue (demostraremos la equivalencia de esta definición con la anterior más adelante):

Definición 3.2.5 *Dada una matriz de confusión normalizada N y su correspondiente matriz de probabilidades marginales M, podemos definir la Kappa de Cohen como:*

$$\kappa = 1 - \frac{\sum_{i,j=1\ldots C}^{i\neq j} N_{i,j}}{\sum_{i,j=1\ldots C}^{i\neq j} M_{i,j}}$$

Para nuestro ejemplo, la matriz M resultante sería la que puede observarse en la Tabla 3.3. Si aplicamos la fórmula de esta última definición, obtendríamos:

$$\kappa = 1 - \frac{(0.\widehat{3} + 0) + (0 + 0) + (0{,}1\widehat{6} + 0)}{(0{,}25 + 0{,}08\widehat{3}) + (0{,}0\widehat{5} + 0{,}02\widehat{7}) + (0.\widehat{1} + 0{,}1\widehat{6})} = 1 - 0{,}72 = 0{,}28$$

		Predicción			Suma real
		-1	0	1	
Realidad	-1	$0{,}1\widehat{6}$	0,25	$0{,}08\widehat{3}$	0.5
	0	$0{,}0\widehat{5}$	$0{,}08\widehat{3}$	$0{,}02\widehat{7}$	$0{,}1\widehat{6}$
	1	$0.\widehat{1}$	$0{,}1\widehat{6}$	$0{,}0\widehat{5}$	$0.\widehat{3}$
Suma predicción		$0.\widehat{3}$	0.5	$0{,}1\widehat{6}$	1

Tabla 3.3: M para el ejemplo de análisis de sentimiento

Tanto la matriz normalizada N como la matriz de distribuciones marginales M pueden verse como matrices de probabilidades. En el primer caso, $N_{i,j}$ representa la probabilidad de que un elemento sea clasificado como de clase j cuando en realidad es de clase i. En el segundo $M_{i,j}$ representa la probabilidad de que un elemento cualquiera tenga clase real i y otro cualquiera tenga clase predicha j (en la definición 3.2.4 esto se traduce en un producto al tratarse se sucesos independientes).

En ambos casos las matrices representan exhaustivamente todos los casos, por lo que se verifican las siguientes propiedades de forma inmediata:

Propiedad 3.2.1

1. *Sea N una matriz de confusión normalizada. Entonces $sum(N) = 1$.*
2. *Sea M una matriz de probabilidades marginales. Entonces $sum(M) = 1$.*

Es fácil comprobar que las definiciones 3.2.1 y 3.2.5 son en efecto equivalentes.

Proposición 3.2.2 *Las definiciones 3.2.1 y 3.2.5 son equivalentes*

Demostración
Llamemos κ_1 a la κ de la definición 3.2.1 y κ_2 a la κ de la definición 3.2.5. Tenemos entonces, sustituyendo las definiciones de *cc* en el denominador 3.2.1:

$$\kappa_1 = \frac{co - cc}{1 - cc} = \frac{co - cc}{1 - \sum_{i=1}^{C}(N_{\bullet i} \cdot N_{i\bullet})} \tag{3.1}$$

y operando en la definición 3.2.5

$$\kappa_2 = 1 - \frac{\sum_{i,j=1\ldots C}^{i\neq j} N_{i,j}}{\sum_{i,j=1\ldots C}^{i\neq j} M_{i,j}} = \frac{\sum_{i,j=1\ldots C}^{i\neq j} M_{i,j} - \sum_{i,j=1\ldots C}^{i\neq j} N_{i,j}}{\sum_{i,j=1\ldots C}^{i\neq j} M_{i,j}} \tag{3.2}$$

Vamos a probar que $\kappa_1 = \kappa_2$ probando la igualdad entre numeradores y entre denominadores. El denominador de la expresión (3.1) se puede reescribir usando la definición de matriz de probabilidades marginales 3.2.4, como:

$$1 - \sum_{i=1}^{C}(N_{\bullet i} \cdot N_{i\bullet}) = 1 - \sum_{i=1}^{C} M_{i,i}$$

Ahora bien, 1 es la suma de la matriz M completa (propiedad 3.2.1) y si le quitamos la suma de la diagonal queda la suma del resto de los elementos, que es justo el denominador de la expresión (3.2).

Para la igualdad de numeradores consideramos en primer lugar el numerador de la expresión 3.2. Restamos y sumamos 1:

$$\sum_{i,j=1\ldots C}^{i\neq j} M_{i,j} - 1 + 1 - \sum_{i,j=1\ldots C}^{i\neq j} N_{i,j} = (\sum_{i,j=1\ldots C}^{i\neq j} M_{i,j} - 1) + (1 - \sum_{i,j=1\ldots C}^{i\neq j} N_{i,j}) \tag{3.3}$$

El primer sumando de esta expresión suma todos los elementos de M menos su diagonal y a continuación le resta 1. Como 1 es la suma de todos los elemento de M (propiedad 3.2.1) el resultado de esta diferencia será la suma de la diagonal en negativo, lo que teniendo en cuenta la definición alternativa de *cc* (definición de la proposición 3.2.1) nos da:

$$\sum_{i,j=1\ldots C}^{i\neq j} M_{i,j} - 1 = -\sum_{i=1}^{C} M_{i,i} = -cc \tag{3.4}$$

El segundo componente del numerador expresado como indica 3.3 es 1 menos la suma de todos los elementos de la matriz normalizada N menos su diagonal. De nuevo por la propiedad 3.2.1 tenemos que la suma completa es 1, por lo que al quitar todos los

elementos menos la diagonal nos queda solo la diagonal de N, que, según la definición 3.2.2 corresponde con co. Por tanto, a partir de la expresión (3.3), el numerador de κ_2 se reduce a $-cc + co$, lo que corresponde al numerador de la definición de κ_1 en (3.1).

Por tanto, tanto el numerador como el denominador coinciden y $\kappa_1 = \kappa_2$ ■

3.3. Kappa con Pesos

Sin embargo, Kappa también presenta un problema que puede afectar a nuestra evaluación: cuando hay más de dos etiquetas y las variables mantienen un orden y una separación entre etiquetas aproximadamente igual (esto es, variables de tipo intervalo o ratio, ver 2.4.1) Kappa no tiene en cuenta la "distancia" entre el valor predicho y el valor real. Pensando en nuestro ejemplo de la tabla 3.1, esto significaría que clasificar un texto realmente positivo como si fuese neutro es igual de "malo" que clasificarlo como negativo, cuando en realidad no debería ser así; al clasificar como neutro se estaría acercando más al acierto que al clasificarlo como negativo.

La métrica Kappa con Pesos (*Weighted Kappa*), que denotaremos como $w\kappa$, tiene en cuenta este matiz para evaluar de una forma más realista un modelo de clasificación con más de dos etiquetas sobre variables de tipo intervalo o ratio.

Su cálculo es muy sencillo, ya que únicamente necesitamos definir una nueva matriz: la matriz de pesos, que denominaremos W. Esta matriz W tiene la misma forma que la matriz F y se rellenará con los pesos (distancias) entre una etiqueta y otra, esto podría verse como la importancia de fallo entre la etiqueta predicha y la etiqueta real. Normalmente, los pesos están distribuidos como se observa en la Tabla 3.4, la diagonal rellena de ceros y el resto de celdas aumentando el valor en uno por cada celda alejada de la etiqueta real.

		Predicción					
		etiqueta1	etiqueta2	...	etiqueta x	...	etiquetaC
Realidad	etiqueta1	0	1	...	|1 - x|	...	C - 1
	etiqueta2	1	0	...	|2 - x|	...	C - 2
	...	...	...	...	...	...	...
	etiqueta x	x - 1	x - 2	...	0	...	C - x
	...	...	...	...	...	...	...
	etiquetaC	C - 1	C - 2	...	C - x	...	0

Tabla 3.4: Matriz de pesos W

Ya con esta matriz podemos calcular $w\kappa$, se utiliza la misma fórmula que el cálculo de κ pero añadiendo el valor de los pesos contenidos en W.

Definición 3.3.1 *Dada una matriz de confusión normalizada N, su correspondiente matriz de probabilidades marginales M y una matriz de pesos W, se define la Kappa de Cohen con peso como:*

$$w\kappa = 1 - \frac{\sum_{i,j=1}^{C} W_{i,j} \cdot N_{i,j}}{\sum_{i,j=1}^{C} W_{i,j} \cdot M_{i,j}}$$

Para nuestro ejemplo de análisis de sentimiento tendríamos la matriz W de la Tabla 3.5, con la que, tras aplicar la fórmula, obtendríamos $w\kappa = 0{,}25$

		Predicción		
		-1	0	1
Realidad	-1	0	1	2
	0	1	0	1
	1	2	1	0

Tabla 3.5: W para el ejemplo de análisis de sentimiento

Otra métrica que parte de esta última es la llamada *Quadratic Weighted Kappa* (Kappa con Pesos Cuadráticos), que denotaremos $qw\kappa$. En realidad se trata de la métrica $w\kappa$ pero, en lugar de utilizar un peso igual al valor absoluto de la diferencia de las etiquetas (ver Tabla 3.4), el peso a incluir en cada celda cuya fila es i y cuya columna es j de la matriz W se calcularía de la siguiente manera:

$$W_{i,j} = \frac{(i-j)^2}{(C-1)^2}$$

Una vez obtenida la matriz W de pesos calculada de la forma anterior, se utilizaría la fórmula definida en 3.3.1. La idea es penalizar más aún las diferencias (de forma cuadrática y no lineal).

3.4. Comportamiento anómalo de Kappa con Pesos y Kappa con Pesos Cuadráticos

Obsérvese que, a partir de la definición 3.2.5 y la definición 3.3.1, si consideramos la matriz de pesos que tiene ceros en la diagonal y unos en el resto de los valores, tenemos que $w\kappa = \kappa$. Sería esperable entonces que si aumentamos el "peso", esto es, si incrementamos los valores de fuera de la diagonal a valores mayores que 1, se tuviera que en general $w\kappa < \kappa$, ya que estamos *castigando* más los errores. Sin embargo, esto no ocurre en varias ocasiones, como se muestra la siguiente matriz de confusión de ejemplo:

2	1	0
1	0	1
2	0	2

Tendríamos un κ = 0,134 y un $w\kappa$ = 0,222, por lo que $w\kappa > \kappa$, pareciendo estar $w\kappa$ *castigando* menos de lo que debería al modelo, incluso lo estaría *premiando*.

Para comprobar si se trata de un caso particular o de lo contrario sucede a menudo, se han realizado experimentos generado varias matrices de confusión aleatorias y se ha comprobado que en varias ocasiones se da este comportamiento no deseado. Si bien es cierto que, a valores mayores de valor de κ (cuando el modelo es de mayor calidad), parece que este comportamiento de $w\kappa$ se da en un menor número de ocasiones, pero sigue siendo muy significante.

Para intentar corregir o al menos minimizar esta situación indeseable definimos nuestra propia variante de Kappa.

3.5. Kappa Penalizada

3.5.1. Definición de Kappa Penalizada

La medida *Penalized Kappa* o *Kappa Penalizada*, que a partir de ahora denotaremos como $p\kappa$, consiste en una variante de la Kappa de Cohen κ, y constituye una de las aportaciones de este trabajo. Para su comprensión, es necesario definir en primer lugar la que llamaremos *matriz de confusión penalizada*, que denotaremos como P. Esta matriz P se obtiene a partir de la matriz de confusión F multiplicando el valor de cada celda por el valor de la penalización (distancia entre etiquetas). Formalmente tendríamos:

Definición 3.5.1 *Dada una matriz de confusión F:*

1. *Definimos la matriz penalizada P asociada a F como:*

$$P_{i,j} = F_{i,j} \cdot distancia(i,j)$$

donde:

$$distancia(i,j) = \begin{cases} |i-j| & \text{si } i \neq j \\ 1 & \text{si } i = j \end{cases}$$

2. *Definimos la medida Kappa Penalizada, $p\kappa$ de F como la Kappa de la matriz P asociada a F.*

Para una mayor comprensión de $p\kappa$, utilizaremos el ejemplo anterior de análisis de sentimiento en textos. A partir de la matriz de confusión F para el ejemplo, que recordamos es:

		Predicción			Suma real
		-1	0	1	
Realidad	-1	1	2	0	3
	0	0	1	0	1
	1	1	0	1	2
Suma predicción		2	3	1	6

Siguiendo la definición anterior, en primer lugar generamos la matriz de confusión penalizada P (ver Tabla 3.6). A continuación para obtener la medida κ de P que será la $p\kappa$ de F debemos calcular en primer lugar la matriz normalizada N_p de P (ver Tabla 3.7).

		Predicción			Suma real
		-1	0	1	
Realidad	-1	1	2	0	3
	0	0	1	0	1
	1	2	0	1	3
Suma predicción		3	3	1	7

Tabla 3.6: P para el ejemplo de análisis de sentimiento 3.1

		Predicción			Suma real
		-1	0	1	
Realidad	-1	0,1429	0,286	0	0,4289
	0	0	0,1429	0	0,1429
	1	0,286	0	0,1429	0,4289
Suma predicción		0,4289	0.4289	0,1429	1

Tabla 3.7: N_p para el ejemplo de análisis de sentimiento

Ahora, utilizando la definición 3.2.1 obtenemos:

$$co = 0{,}1429 + 0{,}1429 + 0{,}1429 = 0{,}4287$$

$$cc = 0{,}4289 \cdot 0{,}4289 + 0{,}4289 \cdot 0{,}1429 + 0{,}1429 \cdot 0{,}4289 = 0{,}30653$$

$$p\kappa = \frac{0{,}4287 - 0{,}30653}{1 - 0{,}30653} = 0{,}17617$$

3.5.2. Relación entre Kappa y Kappa Penalizada

En la sección 3.4 hemos visto que la métrica Kappa con Pesos se comportaba de forma anómala en el sentido de que, aunque esperábamos que fuera menor que Kappa al penalizar los errores más alejados, en la práctica no era así. Nos queda por comprobar si nuestra métrica, la Kappa Penalizada se comporta mejor en este aspecto.

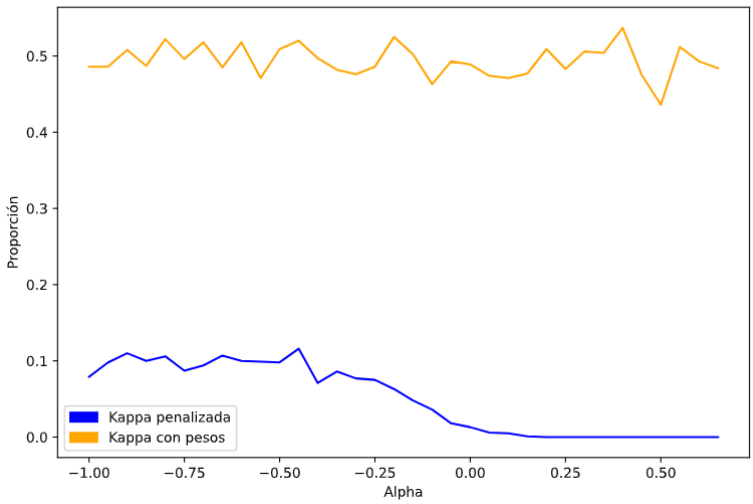

Figura 3.1: Comparación del comportamiento entre Kappa Penalizada y Kappa con Pesos para distintas cotas α de κ en matrices de confusión generadas al azar
Fuente: Elaboración propia

Para ello hemos llevado a cabo experimentos generando matrices de confusión al azar de tamaño 3 y hemos comparado el valor de κ, $w\kappa$ y $p\kappa$. Además para estos experimentos hemos considerado un parámetro α, que sirve como cota inferior de κ: es decir dado un α concreto, nos fijamos en todas las matrices generadas al azar que verificar que $\kappa > \alpha$. De entre todas estas matrices calculamos qué proporción media de de ellas verifica que $p\kappa > \kappa$ y qué proporción media verifica que $w\kappa > \kappa$. El valor α se mueve desde -1 hasta 0.7 con saltos de de 0.1 en 0.1. La gráfica 3.1 muestra el resultado con los valores medios obtenidos

tras generar 10 000 matrices de confusión con $\kappa > \alpha$ para cada α. Podemos extraer dos conclusiones de esta gráfica:

1. Tal y como adelantábamos, la Kappa con Pesos supera a Kappa a menudo, en concreto aproximadamente el 50 % de las veces, lo que denominamos comportamiento anómalo.

2. Además este comportamiento de $w\kappa$ no depende del valor concreto de κ. En la figura esto se observa porque la gráfica de $w\kappa$ es prácticamente horizontal, y por tanto independiente de κ.

3. La Kappa Penalizada sin embargo, tiene una proporción mucho menor de valores en los que supera a Kappa, siempre por debajo del 15 % de los casos de media.

4. Además, la Kappa Penalizada tiende a 0 al considerar valores de Kappa mayores, cosa que como hemos visto no sucede con $w\kappa$. Esto es especialmente interesante porque no indica que para valores de Kappa altos, como los que a menudo se obtienen en *machine learning*, tendremos que prácticamente nunca la Kappa Penalizada supera a Kappa.

En el resto del trabajo utilizaremos $p\kappa$ como medida para decidir cuando un experimento obtiene mejores resultados que otros, aunque también incluiremos Kappa y Kappa con Pesos por ser las más habituales en trabajos de aprendizaje automático.

3.6. Curvas de aprendizaje

Otra técnica de visualización que nos será útil más tarde para evaluar el grado de aprendizaje de la técnica de *machine learning* utilizada es la representación de las curvas de aprendizaje de cada uno de los conjuntos. Las curvas de aprendizaje representan la relación entre cuánto de bueno es el modelo (utilizando una métrica) y la cantidad de datos que conforman el conjunto, es decir, la cantidad de ejemplos que se tienen para entrenar el modelo. Para crear estas curvas se ejecuta el algoritmo de *machine learning* repetidas veces, pero cada una de las veces con un tamaño distinto de datos de entrenamiento (Ng17).

Por ejemplo el algoritmo podría comenzar con un conjunto de entrenamiento correspondiente de tamaño el 10 % del tamaño total del conjunto de datos, y aumentar en cada iteración el tamaño del conjunto en un 5 % hasta terminar con una ejecución con el 80 % de los datos. En cada una de las iteraciones se realizan las correspondientes predicciones con los datos del conjunto de *test* (que sería el restante tras tomar el porcentaje de conjunto de *train* de esa iteración) y se calcula la métrica que refleja la bondad del modelo, que en nuestro trabajo será Kappa con Pesos. Así, para cada iteración tendríamos un porcentaje de datos de entrenamiento y su correspondiente valor de métrica. Normalmente para cada iteración se realiza la predicción un cierto número de veces y se toma el valor medio con la intención de obtener un valor de la métrica más riguroso.

Se denominan "curvas" y no "curva" porque es usual que no solo se realice en cada iteración la predicción del conjunto de *test*, si no también del propio conjunto de *train* con el que se entrena el modelo. Esto es útil ya que puede ayudar a comprender mejor la representación visual que se obtiene.

Una vez obtenidos los resultados del algoritmo se realiza su representación visual, que normalmente corresponde a una gráfica donde el eje de abscisas representa el tamaño del conjunto de entrenamiento y el eje de ordenadas el valor de la métrica utilizada (ver Figura 3.2).

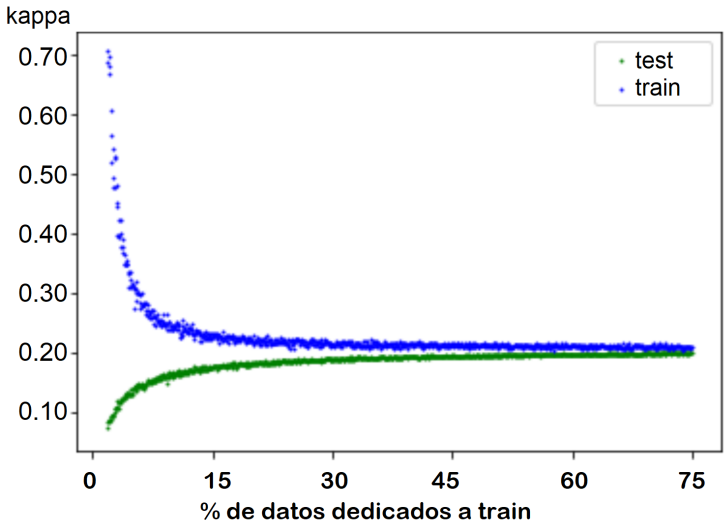

Figura 3.2: Ejemplo de curva de aprendizaje
Fuente: Elaboración propia

Si analizamos la curva de *test* (en verde) del ejemplo de la Figura 3.2 es trivial que a medida que aumenta el tamaño del conjunto de entrenamiento, el valor de la métrica κ aumenta, el modelo mejora, es decir, al entrenar con más valores el modelo predice mejor la etiqueta de valores nuevos. Aunque podría parecer lógico pensar que esto también ocurriera para la curva de *train* (en azul en la figura), no es así; para porcentajes más altos de datos dedicados al conjunto de entrenamiento el valor de la métrica va a decrecer. Esto se debe a que el modelo, al tener cada vez más elementos de entrenamiento va generalizando mejor y deja de "memorizar" los valores, por lo que a mayor porcentaje de datos de entrenamiento mayor será el número de fallos y menor será κ, que sin embargo siempre es mejor que en el caso del test. Para comprender esto mejor supongamos que en la primera iteración del algoritmo el conjunto de datos de entrenamiento está comprendido por únicamente tres textos: uno etiquetado como positivo, otro como negativo y otro como neutro; en este caso la predicción del conjunto de entrenamiento sería muy fácil, ya que existe un único ejemplo de cada uno y se va a predecir esos mismos ejemplos, es muy fácil que el modelo los haya aprendido. Si por el contrario tenemos 100 ejemplos más, variados en etiquetas, quizá algunos de ellos "confunden" al modelo, lo hacen más generalista, ha dejado de "memorizar" los ejemplos para pasar a crear relaciones o patrones útiles para la predicción, por lo que al predecir, aunque sea sobre los propios datos de entrenamiento, comete más errores.

Para llevar a cabo la interpretación de estas curvas debemos saber que:

- Si la curva de *test* tiende a un valor constante cuando se aumenta el tamaño de *train* (como ocurre en el ejemplo de la Figura 3.2) es poco probable que el modelo vaya a mejorar introduciendo nuevos datos al conjunto de entrenamiento. Pero para asegurarnos de esto es necesaria la visualización de la curva de *train* conjunto a ella, si a esta curva le ocurre lo mismo y además la distancia entre ambas curvas es corta, podemos estar seguros de que la introducción de nuevos datos no va a mejorar el modelo.

- Por el contrario, si la distancia entre ambas es grande y además la curva de *test* no se mantiene, si no que toma cada vez valores más altos, es muy probable que al introducir nuevos datos al conjunto de entrenamiento el modelo mejore.

Capítulo 4

Experimentos

En este capítulo se muestran y se discuten los resultados obtenidos de los experimentos realizados con las técnicas utilizadas: *IBM Watson* y nuestro Método Sencillo. En primer lugar (Sección 4.1) se describe la metodología que se ha seguido para llevar a cabo los experimentos del siguiente apartado, más tarde (Sección 4.2) se comparan los resultados obtenidos utilizando ambas técnicas y teniendo en cuenta las diferencias entre los dos tipos de ejemplos (tipo competitivo y tipo reseña), para describir seguidamente (Sección 4.4) las posibles explicaciones de las notables diferencias encontradas. A continuación (Sección 4.3) se discute la diferencia de magnitud entre las métricas obtenida en los resultados anteriores. En los apartados finales se analizan algunos de los factores que pueden explicar estos resultados: la complejidad de los textos (Sección 4.5), la presencia de elementos de *Twitter* (Sección 4.6) y la influencia de las palabras mal escritas (Sección 4.7).

4.1. Metodología

En primer lugar se han realizado los experimentos utilizando el servicio de análisis de sentimiento de *IBM Watson*. Para cada uno de los seis conjuntos definidos en el apartado 2.4.2 se han hecho llamadas a la API de *IBM Watson* con cada uno de los textos y se ha almacenado su resultado. Dado que el sentimiento resultante se obtiene en lenguaje natural, este se ha transformado al formato de etiquetas numéricas siguiendo la correspondencia *negative* $\mapsto$ -1, *neutral* $\mapsto$ 0 y *positive* $\mapsto$ 1, ya que {-1, 0, 1} son las etiquetas empleadas en los conjuntos etiquetados manualmente. Sin embargo, para el conjunto *reseñas-amazon* el conjunto etiquetado manualmente tiene las etiquetas reales {1,2,3,4,5} (la calificación dada por el usuario para el producto, de una a cinco estrellas). En este caso no se ha podido utilizar el sentimiento en lenguaje natural devuelto por Watson, sino que se ha utilizado otra característica que da el servicio: el *score* o puntuación, y que representa el sentimiento como un número real comprendido entre -1 y 1. A partir de este valor se han generado intervalos que han permitido convertir estos números reales a cada una de las etiquetas: 1 para $-1 \leq score \leq -0{,}6$, 2 para $-0{,}6 < score \leq -0{,}2$, 3 para $-0{,}2 < score \leq 0{,}2$, 4 para $0{,}2 < score \leq 0{,}6$ y 5 para $0{,}6 < score \leq 1$, es decir, saltos de 0,4 por cada etiqueta, correspondientes a dividir un intervalo de longitud 2 (el [-1,1]) en cinco partes.

Para los experimentos realizados utilizando el Método Sencillo basado en *Random Forest* se han seguido lo siguientes pasos con cada uno de los conjuntos definidos en 2.5:

1. *Generar la matriz de apariciones.* En primer lugar se ha obtenido un conjunto que incluye todas las palabras contenidas en el total de textos, sin repeticiones y ex-

cluyendo *stop words* (pronombres, proposiciones, verbos modales...). Después, para cada texto se ha generado un vector de apariciones de las palabras del conjunto anterior. Finalmente, tendríamos una matriz cuyas columnas son el conjunto de palabras totales en los textos, cuyas filas son los textos y cuyos valores son las apariciones. Supongamos que tenemos un conjunto de ejemplo de tamaño tres con reseñas de un restaurante italiano, en la Tabla 4.1 podríamos observar la forma que tendría la matriz de apariciones final. Obsérvese que el conjunto de palabras {los, a, mi, le, han, estaban, muy, no} no pertenecen al conjunto de columnas, sino que pertenecen al conjunto de *stop words*.

2. *Etiquetado.* A la matriz anterior se le añade una nueva columna, la etiqueta o *label*, que toma los valores -1, 0, 1 según el mensaje contenga un sentimiento negativo, neutro o positivo en la clasificación manual, (o {1,2,3,4,5} en el conjunto *amazon-4000*).

3. *Modelización y predicción.* Una vez se tiene la matriz del punto anterior, se divide el conjunto (vectores de apariciones de palabras obtenidos de la matriz y etiquetas reales) en dos: un conjunto de *train* o entrenamiento y un conjunto de *test* o prueba, en concreto se ha utilizado un 60 % de los datos para *train* y un 40 % de los datos para *test*. Después de esto, se ha entrenado el modelo basado en *Random Forest* con el conjunto de *train* y se han realizado las predicciones para cada uno de los textos del conjunto de *test*. Todo este proceso de modelización y predicción se ha realizado un número determinado de veces, en concreto 1000, almacenando en cada iteración el resultado de las métricas κ, $qw\kappa$ y $p\kappa$.

4. *Métricas.* Finalmente se obtienen los valores medios de κ, $qw\kappa$ y $p\kappa$ de las 1000 predicciones realizadas. Los valores de estas métricas serán los utilizados para la valoración de resultados.

	espaguetis	buenos	camareros	ausentes	hermano	gustado	salados
1*	1	1	0	0	0	1	0
2*	0	0	1	1	0	0	0
3*	1	0	0	0	1	1	1

Tabla 4.1: Ejemplo de matriz de apariciones
Fuente: Elaboración propia

1* Los espaguetis estaban muy buenos, me han gustado
2* Los camareros estaban ausentes
3* A mi hermano no le han gustado los espaguetis, estaban salados

El código desarrollado y que sigue la metodología anterior para la obtención de los resultados que se muestran en los siguientes apartados se puede encontrar en `https://github.com/raqblanc/analsis-sentimiento-contexto-competitivo`.

Nótese que este tratamiento es extremadamente simple y que evita muchos pasos comunes en el tratamiento de textos tales como la eliminación de palabras poco o demasiado frecuentes o la extracción de raíces sintácticas (ver (Agg18) para una descripción de estos pasos adicionales). Estos aspectos, así como muchos otros como el ajuste fino de hiperparámetros en el modelo, o la prueba con otros métodos diferentes a *Random Forest* se han evitado a propósito porque de lo que se trata aquí no es de competir con *IBM Watson* sino

de observar si las diferencias entre ambos métodos varían según el tipo de ejemplo, y si es así determinar posibles explicaciones.

4.2. Comparativa entre Watson y el Método Sencillo para los dos tipos de ejemplos

Los valores de las métricas que se han obtenido para cada uno de los tres conjuntos utilizados en este trabajo con tipo de texto competitivo y con tipo de texto no competitivo son los que pueden observarse en las Tabla 4.2 y 4.3, respectivamente. En ambas tablas está marcado en amarillo el valor de $p\kappa$ mayor entre el obtenido por el Método Sencillo o por el servicio *IBM Watson*, es decir, el método que nos proporciona mejores resultados para cada conjunto.

		κ	$qw\kappa$	$p\kappa$
trump-2020	M. sencillo	0.145597	0.141955	0.148224
	Watson	0,149467	0,292411	0,137433
trump-2016	M. sencillo	0,303935	0,36889	0,270191
	Watson	0,203099	0,227255	0,150332
biden-2020	M. sencillo	0,137738	0,13357	0,140985
	Watson	0,121397	0,217956	0,109454

Tabla 4.2: Resultados obtenidos con los conjuntos de textos competitivos (elecciones). En amarillo el método que obtiene mejores resultados. *Fuente: Elaboración propia*

		κ	$qw\kappa$	$p\kappa$
aerolinea-american	M. sencillo	0.358197	0.487246	0.273658
	Watson	0.428818	0.518163	0.32465
aerolinea-delta	M. sencillo	0,416639	0,503547	0,364994
	Watson	0.492358	0.615735	0.438678
amazon-4000	M. sencillo	0,120598	0,237279	0.003087
	Watson	0.242556	0.54257	0.10116

Tabla 4.3: Resultados obtenidos con los conjuntos de textos no competitivos (reseñas). En amarillo el método que obtiene mejores resultados. *Fuente: Elaboración propia*

Para los conjuntos utilizados del primer tipo (competitivos) los valores de $p\kappa$ del Método Sencillo son siempre mayores que los valores de $p\kappa$ obtenidos por Watson, es decir el modelo creado con el Método Sencillo da mejores resultados para estos tipos de texto.

Por otro lado, para los conjuntos utilizados del segundo tipo (no competitivos) los valores de $p\kappa$ son siempre mayores para el caso de Watson, por lo que para este tipo de textos parece que el método más conveniente a utilizar para el análisis de sentimiento es Watson.

Los resultados además quedan parcialmente confirmados por los valores de las otras dos medidas: κ y $qw\kappa$, salvo en dos ocasiones: en *trump-2020* y en *biden-2020*, en los que, si nos fijamos en la medida $qw\kappa$ llegaríamos a la conclusión de que Watson obtiene mejores resultados. En el ambos casos, la explicación viene dada por lo que mencionamos en la sección 3.5.2; estamos valores anómalos de $qw\kappa$ en los que $qw\kappa > \kappa$ y además por una distancia notable en el caso de la medida para Watson. Fue por esta razón, pensando que

en general era preferible que la versión con peso o penalizada fuera menor que κ por la que introdujimos nuestra medida $p\kappa$. Hay que mencionar que en el caso *biden-2020* se tiene también que para el Método Sencillo $p\kappa > \kappa$, pero por un estrecho margen.

En todo caso, como mencionábamos en la introducción, el objetivo no es "derrotar a Watson", sino observar si hay diferencia entre el rendimiento de Watson comparado con el Método Sencillo según el tipo de experimento. Los resultados parecen sugerir que tal diferencia existe, señalando que entre experimentos basados en eventos competitivos, en este caso elecciones, funciona mejor el método que utiliza conjuntos etiquetados a mano para ese evento en concreto, el llamado Método Sencillo, con respecto a los ejemplos basados en opiniones generales, donde la biblioteca Watson parece claramente superior.

Un detalle que merece la pena mencionar es el anormalmente bajo valor de $p\kappa$ para el Método Sencillo en el caso *amazon-4000*, donde el valor se aproxima a 0. Para explicarlo debemos recordar que en este ejemplo teníamos no 3 valores (negativo, neutro, positivo), sino 5 valores, por lo que los valores penalizados pueden ser menores al ser mayor la distancia entre el valor correcto y el predicho, es decir pesar más la penalización.

4.3. Margen de mejora del Método Sencillo

En todo caso parece que los valores de las métricas obtenidos son muy bajos, estando en casi todos los casos por debajo 0.5. Aunque no sea el objetivo primordial del trabajo, cabe preguntarse si en particular para el Método Sencillo sería posible mejorarlos, quizás con datasets más grandes.

Con este objeto hemos recurrido a las curvas de aprendizaje que, como se indicaba en el apartado 3.6, resultan útiles para comprender el aprendizaje del modelo y determinar cuánto puede mejorar este con la introducción de nuevos datos al conjunto. Es por esto que se han realizado las curvas de aprendizaje del Método Sencillo para cada uno de los seis conjuntos de datos. Como puede observarse en cada una de las gráficas de la Figura 4.1, se tienen unas curvas de aprendizaje de *test* crecientes en la mayoría de casos aunque en algunos de ellos parece tender ya hacia una asíntota. Observamos además con una gran distancia entre la curva de *text* y la curva de *train*, cuando vimos que lo normal es que convergieran. Además lo que es muy inusual es que la curva de entrenamiento no desciende, o apenas empieza a hacerlo. Todo esto lo que nos está indicando que la introducción de nuevos ejemplos a los conjuntos podría mejorar el modelo teniendo unos valores de $p\kappa$ mayores, aunque no podemos descartar el otro escenario posible: que aunque tengamos datasets mayores la curva de test ya no crezca más, estabilizándose y la de entrenamiento sea la que poco a poco converja hacia ella.

Esto es explicable porque como hemos dicho el Método Sencillo trabaja con vectores de palabras. Aunque el número de palabras en un idioma es limitado, se trata de una cantidad elevada comparada con nuestros conjuntos de entrenamiento, que solo alcanzan unos pocos cientos de filas. Por tanto aunque el sistema se "aprende" las palabras del conjunto de entrenamiento (gráfica naranja) falla ante las de *test* porque muchas son nuevas.

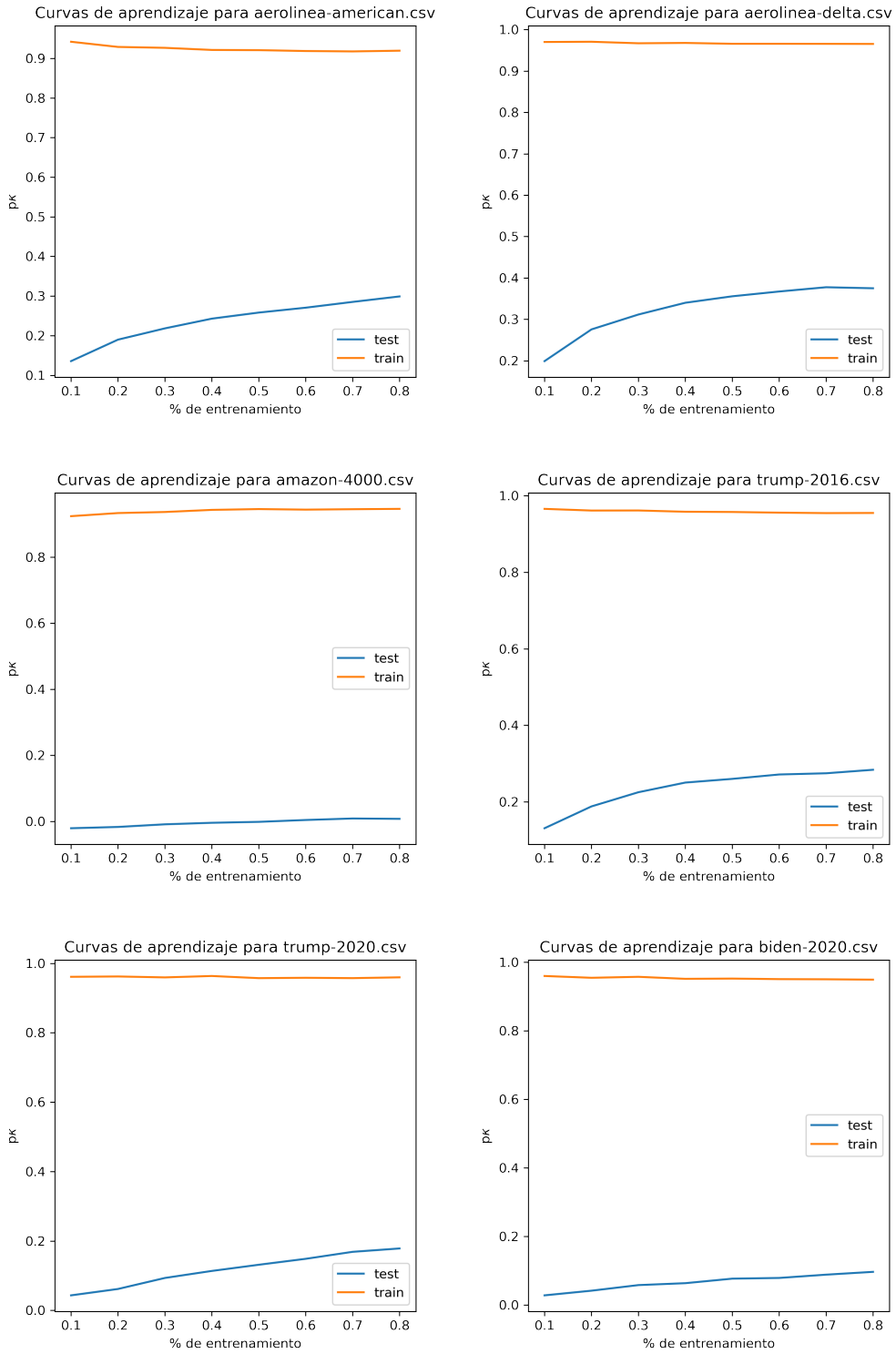

Figura 4.1: Curvas de aprendizaje
Fuente: Elaboración propia

4.4. Posibles explicaciones de las diferencias entre el Método Sencillo y Watson en los diferentes tipos de ejemplos

Es difícil asegurar cuál es la explicación de esta diferencia de comportamiento entre Watson y el Método sencillo según el tipo de ejemplo. Dos posibles explicaciones son:

1. La idea original que generó este trabajo: que en los eventos competitivos el sentimiento depende enormemente del contexto de forma que una expresión que puede ser positiva en un caso será negativa en otro, dependiendo de explicaciones ajenas al propio texto.

   Un ejemplo: la palabra *deplorables* tiene un significado netamente negativo en inglés. En un momento de la campaña *trump-2016* la candidata H. Clinton utilizó esta palabra para referirse a algunas mujeres que apoyaban a D. Trump. Sorprendentemente algunas seguidoras de Trump tomaron esta palabra por bandera, haciendo incluso camisetas con el lema *Deplorables for Trump*. En este caso, Watson al inferir el sentimiento de esta expresión con toda probabilidad deducirá que se trata de un *tweet* negativo hacia Trump. Sin embargo, la persona que clasifica manualmente y que conoce el contexto marcará el *tweet* como lo que realmente es: positivo al venir de un grupo de apoyo a Trump. Además a esta dependencia del contexto obliga la propia red social, *Twitter*, al limitar el número de caracteres, lo que obliga al usuario a sintetizar y referirse, implícitamente, al contexto sin mencionarlo por falta de espacio, utilizando recursos como hashtags, urls externas, menciones, etc. En el caso de las opiniones los textos suelen más largos y autocontenidos, lo que quizás beneficie a Watson que es capar de un análisis más profundo que nuestro Método Sencillo. También puede ayudar a "confundir" a Watson la existencia de más de un actor en los mensajes en los contextos competitivos. Aunque al llamar a la biblioteca indicamos explícitamente en quién debe fijarse resulta sencillo que no sepa si una expresiónp positiva vaya destinada a uno u otro candidato.

2. Una segunda explicación es que no se trate de diferencias semánticas dadas por el contexto sino de diferencias de otro tipo: quizás las frases en uno de los casos están peor construidas, lo que puede confundir a Watson, o tengan un lenguaje a un nivel diferente (más sencillo o más complejo).

4.5. Complejidad de los textos

Como se indicaba en la segunda explicación del aparatado 4.4, una de las posibles razones por las que los resultados sean mejores o peores para las dos técnicas utilizadas y entre los dos distintos tipos de texto podría ser la complejidad del texto, que podría repercutir en el análisis de sentimiento. Para poder llegar a una conclusión sobre este punto se han realizado comprobaciones de complejidad de texto con la biblioteca *textstat* de *Python* (Ban) que se utiliza sobre todo para ver su un texto es conveniento para la lectura de un niño de una determinada edad o grado escolar. Se han utilizado las siguientes medidas, cuyos resultados pueden observarse en la Tabla 4.4:

1. La fórmula de Dale–Chall Readability (DC48) proporciona un indicador de la dificultad de comprensión de un texto, por lo que a mayor valor resultante de la fórmula, mayor será la complejidad. Se define como:

$$0{,}1579 \cdot \left(\frac{palabras\_dificiles}{total\_palabras} \cdot 100 \right) + 0{,}0496 \cdot \frac{total\_palabras}{total\_oraciones}$$

donde *palabras_dificiles* es el número total de palabras en el texto que no están contenidas en una tabla de "palabras fáciles", esta tabla que contiene 3000 palabras del inglés consideradas sencillas de comprender por estudiantes americanos de cuarto grado.

2. Text Standard tiene en cuenta los resultados de diferentes fórmulas de la biblioteca *textstat* y estima el grado escolar necesario para entender el texto. Por lo tanto un mayor valor se traduce como mayor complejidad del texto.

3. La fórmula de Flesch Reading Ease (Fle14) indica la facilidad de lectura de un texto, por lo que a mayor valor resultante de la fórmula, menor será la complejidad, se define como:

$$206{,}835 - 1015 \cdot \left(\frac{total\_palabras}{total\_oraciones}\right) - 84{,}6 \cdot \left(\frac{total\_silabas}{total\_palabras}\right)$$

		Dale-Chall ↑	Text Standard ↑	Flesch ↓
Competitivos	*trump-2020*	10,56	13,86	18,6
	trump-2016	11,16	15,18	3,4
	biden-2020	10,66	11,71	32,51
No competitivos	*aerolinea-american*	7,38	9,89	63,76
	aerolinea-delta	8,17	8,18	68,81
	amazon-4000	6,33	8,85	68,93

Tabla 4.4: Complejidad de los textos competitivos y no competitivos.
Se han utilizando las fórmulas Dale–Chall Readability, Flesch Reading Ease y Text Standard. La fecha indica en qué dirección aumenta la complejidad (flecha arriba mayor complejidad cuanto mayor sea el valor, flecha hacia abajo cuando valores mayores indican menor complejidad)

Para los resultados de las fórmulas *Dale-Chall*, que indican la dificultad, tenemos valores más altos en los textos competitivos que en los no competitivos, al igual que para *Text Standard* que indica el grado de escolarización, mientras que para los resultados de *Flesch*, que indican la facilidad del texto, tenemos valores más pequeños para los textos competitivos. Dados estos resultados, podríamos deducir que existe una mayor complejidad en los textos de tipo competitivo.

Esto parecería apoyar la segunda propuesta: que no es tanto el contexto el que perjudica a Watson sino la complejidad de los textos. Sin embargo, que Watson funcione peor con textos más complejos resulta algo extraño, ya que es una biblioteca especializada en textos largos y estructurados, mientras que aquí se trata de *tweets* que cuentan con una limitación de caracteres en todos los conjuntos excepto el de las reseñas de Amazon, que por lo general también son breves.

Esto nos lleva a preguntarnos si no serán las características especiales de los *tweets* las que marcan la diferencia, pudiendo estar estas características de alguna forma "engañando" a la biblioteca *textstats* y haciendo ver como palabras complejas lo que en realidad son *hashtags*, *URLs* ...

4.6. Importancia de las características de los *tweets*

Dado que cinco de los seis conjuntos utilizados en este trabajo contienen textos en forma de *tweet*, es relevante analizar cómo afectan las características propias de este tipo de mensajes en la complejidad del texto, ya que sospechamos que las fórmulas utilizadas en el apartado 4.5 para medir la complejidad no tienen en cuenta la aparición de estos elementos o pueden confundirlos con palabras complejas. Para ello vamos diferenciar tres tipos de características o elementos muy utilizados en el textos de un *tweet* y que pueden influir en el análisis de la complejidad:

1. Las etiquetas o *hashtags* utilizadas para etiquetado del *tweet*. Suelen utilizarse para aludir a un acontecimiento que está ocurriendo, por lo que con frecuencia hacen referencia a un contexto que no está implícito en el texto. Esto es muy útil ya que los *tweets* cuentan con una restricción de caracteres de 280; al incluir estos *hashtags* en un *tweet* se hace prescindible escribir explícitamente de qué se está hablando. Por ejemplo, imaginemos el *tweet* «Hoy están jugando como nunca antes #halamadrid», si analizásemos el texto sin la parte de etiqueta: «Hoy están jugando como nunca antes», en primer lugar, no sabríamos a quiénes se está refiriendo el texto, ni sabríamos si es positivo hacia los jugadores o no, solo sabríamos que están jugando como nunca; ahora, teniendo en cuenta el *hashtag* #halamadrid podríamos comprender que se trata de los jugadores del Real Madrid, y puesto que "Hala Madrid" se trata del "grito de guerra" de los aficionados del Real Madrid, podríamos incluso deducir que el *tweet* es positivo hacia los jugadores.

2. Las menciones, escritas con un símbolo @ antes del nombre de usuario, utilizadas para hacer referencia a un usuario de la red social. En este trabajo diferenciaremos en alguna ocasión las menciones en dos tipos: menciones a los candidatos y resto de menciones.

3. Los enlaces externos al *tweet* como noticias, artículos, vídeos... que también pueden incluir la parte de contexto no incluida en el texto del propio *tweet*. Por lo tanto el estudio de los textos o las imágenes incluidos en estos enlaces externos podría resultar muy interesante ya que podría enriquecer las predicciones de análisis de sentimiento. Hay que tener en cuenta que la persona que ha clasificado los mensajes manualmente ha consultado estos enlaces para decidir el sentimiento asociado a los textos, por lo que se trata de información contextual muy relevante.

Para comprobar la influencia que tienen estas características de los *tweets* en el análisis de complejidad hemos vuelto a realizar los mismos experimentos del apartado 4.5 con la biblioteca *textstats*, pero esta vez realizando transformaciones en los textos; en concreto eliminando cada una de estas características (etiquetas, menciones a candidatos, resto de menciones y enlaces externos). Estos nuevos resultados puede se encuentran en la Tabla 4.5 de donde se puede observar lo siguiente:

- La presencia de estas características aumentan los resultados del cálculo de complejidad del texto, ya que la eliminación progresiva de estos elementos hace que la facilidad del texto (*Flesh*) aumente y la dificultad del texto (*Dale-Chall*) y el grado de escolarización (*Standard*) disminuya en todos los casos y bruscamente. Obsérvese que como el conjunto de datos *amazon-4000* no se trataba de *tweets* sino reseñas de Amazon el valor resultante es casi idéntico, ya que no suelen contener estos elementos característicos utilizados en *Twitter*.

- La complejidad tras eliminar todos los elementos característicos de *Twitter* (columna más a la derecha) queda muy similar entre los textos de tipo competitivo y los textos de tipo reseña. Por ejemplo, *Dale-Chall* en los conjuntos de las elecciones pasa de estar entre 10 y 11 a valer entre 5 y 6, quedando igual o incluso peor que los valores de *Dale-Chall* para los conjuntos de textos de tipo reseña.

		original	sin *C*	sin *C@*	sin *C@#*	sin *C@#http*
trump-2020	Flesch	18.608	18.524	43.514	49.845	89.176
	Standard	13.864	13.726	7.967	7.389	6.120
	Dale-Call	10.569	10.537	7.220	6.675	5.223
trump-2016	Flesch	3.401	3.402	32.167	35.308	72.601
	Standard	15.185	15.185	11.568	9.305	6.147
	Dale-Call	11.163	11.162	8.533	7.833	5.835
biden-2020	Flesch	32.516	26.983	37.664	40.115	85.964
	Standard	11.713	11.096	8.616	8.330	6.169
	Dale-Call	10.660	9.272	7.398	7.122	5.273
areolinea-american	Flesch	63.762	63.762	75.717	76.619	77.080
	Standard	9.899	9.899	7.502	7.296	7.167
	Dale-Call	7.832	7.832	6.087	5.953	5.885
aerolinea-delta	Flesch	68.812	68.812	75.416	76.632	78.762
	Standard	8.189	8.189	6.950	6.591	6.175
	Dale-Call	8.172	8.172	6.208	5.891	5.527
amazon-4000	Flesch	68.936	68.936	68.939	68.994	68.994
	Standard	8.857	8.857	8.857	8.849	8.849
	Dale-Call	6.338	6.338	6.338	6.335	6.335

Tabla 4.5: Diferencia de complejidad con la eliminación de símbolos propios de *Twitter* En las columnas: *original* se refiere al texto del conjunto sin transformaciones, *sin C* se refiere a la eliminación de la mención en el *tweet* de los candidatos en las elecciones (Trump y Biden), *sin C@* se refiere a lo anterior y además el resto de menciones, *sin C@#* se refiere a lo anterior y además las etiquetas o *hashtags* propios de *Twitter* y *sin C@#http* se refiere a lo anterior y además todos los enlaces externos que contiene el *tweet*.

En resumen, y fijándonos de nuevo en la última columna de la tabla, vemos que al suprimir los elementos propios de *Twitter* la diferencia de complejidad desaparece, y que los valores pasan a ser muy similares entre todos los conjuntos, lo que puede llevarnos a pensar que estos elementos hacen que el cálculo de la complejidad anterior contando con ellos no sea fiable. Las implementación de los 3 métodos empleados no saben interpretar esta simbología; en el cálculo de *Dale-Chall Readaility* el texto de ejemplo utilizado anteriormente: "Hoy están jugando como nunca antes #halamadrid" que contiene un *hashtag*, esta fórmula consideraría como palabra completa "#halamadrid", es más, la consideraría "palabra difícil" al no estar contenida en su tabla de "palabras fáciles", esto mismo ocurriría con las menciones y los enlaces externos, que podrían hacer la fórmula incorrecta en el conteo de sílabas, de palabras y de oraciones.

Cabe preguntarse entonces si estos mismos elementos no estarán marcando la diferencia a la hora de utilizar Watson y el Método Sencillo. Si pensamos en el ejemplo anterior "Hoy están jugando como nunca antes #halamadrid" vemos que es precisamente este "#halamadrid" el que hace que este *tweet* sea positivo para el Real Madrid, y puede que esta

información, obvia para el clasificador humano, no lo sea para Watson.

En consecuencia, podemos intuir que la complejidad lingüística de los textos de tipo competitivo no se diferencia de la de los textos de reseñas, por lo que lo que hace que el Método Sencillo obtenga mejores resultados para textos de tipo competitivo y peores para textos de tipo reseña (como vimos en el capítulo 4.2) no es una complejidad del texto de tipo competitivo que Watson no pueda entender, sino más bien la información semántica o de contexto que aportan estos elementos al clasificador humano. Para comprobar esto, se han vuelto a realizar los experimentos utilizando Watson y el Método Sencillo pero en este caso con los textos transformados, eliminando todos los elementos propios de *Twitter*. De esta manera podremos comprobar tanto si a Watson como al Método Sencillo le afectan estos elementos de tal manera que sus resultados cambien la valoración final de estos.

4.6.1. Frecuencia de elementos específicos de *Twitter* en los ejemplos

Hay que tener en cuenta que de los tres ejemplos de tipo no competitivo dos vienen de *Twitter*, por lo que en principio la presencia de elementos propios de esta plataforma debería ser similar, a no ser que influya la cantidad de elementos utilizados.

Para comprobar este extremo se ha hecho un recuento de apariciones de estos símbolos en los textos (ver Tabla 4.6). Como resultado tenemos que:

- Como era esperable el conjunto *amazon-4000* tiene una media de prácticamente cero elementos de *Twitter* por mensaje.

- De entre los textos provenientes de *tweets*, tenemos que en el caso de los textos de campañas electorales se tiene un número mucho mayor de elementos por *tweet* (entre 3 y 5) mientras que los *tweets* sobre opiniones de aerolíneas tienen de media menos elementos por *tweet*.

A partir de estos resultados podemos decir que es más común el uso en *Twitter* de estos elementos cuando se dan acontecimientos competitivos como procesos electorales, y menos comunes cuando se trata de una reseña o valoración como en el caso de las aerolíneas. Esto puede deberse a lo que comentábamos anteriormente sobre los *hashtags* y el límite de caracteres: el contexto es complejo y cambiante en estos tipos de acontecimientos, por los que el uso de estas etiquetas reduce el tamaño del texto y facilita la comprensión para los lectores que conozcan el contexto que refleja la etiqueta; con un único *hashtag* el lector puede entender el contexto explícito e incluso la opinión de la persona que lo escribió.

	@	*@#*	*@#http*
trump-2020	2.535129	2.836381	3.070259
trump-2016	3.647090	4.545258	5.282860
biden-2020	2.875656	3.132224	3.415936
areolinea-american	1.135918	1.329467	1.37839797
aerolinea-delta	1.1318631	1.414041	1.58865887
amazon-4000	0.002	0.06275	0.063

Tabla 4.6: Cantidad media de elementos por *tweet* en los distintos conjuntos utilizados
Las columnas indican los elementos contados: @ son las menciones, # los *hashtags* y *http* los enlaces externos.

4.6.2. Comparativa entre Watson y el Método Sencillo eliminando elementos específicos de *Twitter*

A continuación, para entender la repercusión de los elementos de *Twitter* en el análisis de sentimiento de los textos de nuestros conjuntos de ejemplo, se han repetido los experimentos con el Método Sencillo y Watson de nuevo cuyos resultados se presentaron en las tablas 4.2 y 4.3 pero esta vez eliminando de los textos estos elementos.

		κ	$qw\kappa$	$p\kappa$
aerolinea-american	Método sencillo	0.356367	0.484860	0.272969
	Watson	0.4270611	0.514842	0.321573
aerolinea-delta	Método sencillo	0.414280	0.498698	0.357710
	Watson	0.485814	0.607978	0.431263
amazon-4000	Método sencillo	0.121836	0.237009	0.002730
	Watson	0.243393	0.542491	0.1015687
trump-2020	Método sencillo	0.132557	0.116170	0.136342
	Watson	0.1400550	0.284566	0.1280756
biden-2020	Método sencillo	0.079216	0.089164	0.087276
	Watson	0.119173	0.2310164	0.106292
trump-2016	Método sencillo	0.265777	0.295872	0.225263
	Watson	0.198890	0.252680	0.149022

Tabla 4.7: Resultados obtenidos utilizando las técnicas Watson y Método Sencillo eliminando de los textos los elementos de *Twitter*

Los nuevos valores de las métricas pueden observarse en la Tabla 4.7, de los que podemos obtener las siguientes ideas:

- Los resultados de los textos de tipo no competitivo apenas cambian con respecto a los resultados en 4.2, esto puede deberse a que en el caso de estos tipos de texto, como vimos en la Tabla 4.6 apenas aparecen estos elementos, y es posible que de aparecer apenas aporten información útil.

- Si observamos los resultados de $p\kappa$ para los textos de tipo competitivo podemos apreciar que la eliminación de estos elementos perjudica más al Método Sencillo que a Watson. Por ejemplo, en *biden-2020* teníamos anteriormente para el Método Sencillo $p\kappa$ = 0,140985 frente al valor $p\kappa$ = 0,087276 que obtenemos tras eliminar estos elementos, mientras que para Watson pasa de $p\kappa$ = 0,109454 (ver Tabla 4.2) a $p\kappa$ = 0,106292, lo que nos hace ver que Watson no se ve tan afectado como el Método Sencillo por estos elementos.

  Cabe destacar que, como puede observarse en la Tabla 4.6, el conjunto *biden-2020* es el segundo entre los seis en el orden de frecuencia de elementos de *Twitter*. Además del alto número de elementos por *tweet* que tiene este conjunto, es posible que para este conjunto estos elementos aporten mucho valor al Método Sencillo, lo que hace que su eliminación haga bajar el valor de $p\kappa$ bruscamente. Obsérvese que en el caso de este conjunto la eliminación de estos elementos incluso provoca que el método Watson proporcione mejores resultados que el Método Sencillo. Este decrecimiento significativo de $p\kappa$ para el Método Sencillo también ocurre con el conjunto *trump-2020* pero en menor medida, la eliminación afecta más al Método Sencillo que a Watson y con bastante diferencia, pero el Método Sencillo sigue obteniendo aun así un valor $p\kappa$ mayor.

- Para *biden-2020* se tiene ahora $p\kappa > \kappa$. Lo mismo sucede para *trump-2020* pero muy poco además el valor de κ es tan bajo que es frecuente que esto le ocurra tanto a $qw\kappa$ como, más rara vez, a $p\kappa$.

Tenemos por tanto que los conjuntos de textos de tipo no competitivo, a pesar de tratarse dos de ellos de *tweets*, apenas se ven afectados por la eliminación de estos elementos, mientras que los conjuntos de textos de tipo competitivo se ven mucho más afectados, incluso llegando a darse en uno de ellos el caso de que el Método Sencillo pierde tanta información que Watson da mejor resultado. Sin embargo en otros dos ejemplos, *trump-2020* y *trump-2016* sigue siendo mejor el Método Sencillo, incluso tras eliminar estos elementos, aunque por menos margen.

4.7. Importancia de palabras mal escritas

Otra posibilidad que debemos considerar, y que podría explicar que Watson actúe peor en los textos de tipo competitivo, puede ser la existencia de palabras mal escritas que, mientras que pueden ser interpretadas por el clasificador humano, quizás el sistema Watson no pueda *comprender*.

Para evaluar esta posibilidad se ha calculado la proporción de palabras mal escritas en cada uno de los textos de ejemplo. Para ello:

- En primer lugar se han eliminado los elementos de *Twitter* mencionados en la sección anterior: *hashtags*, menciones y *URLs* con el fin de que no influyan en el conteo
- A continuación se ha utilizado la biblioteca *pyspellchecker* de *Python* (Bar) que ha permitido obtener el número de palabras mal escritas a través de su funcionalidad de corrección.

	Proporción mal escritas
trump-2020	0.021600
trump-2016	0.043100
biden-2020	0.022000
areolinea-american	0.0396
aerolinea-delta	0.0372
amazon-4000	0.0117

Tabla 4.8: Proporción de palabras mal escritas del total de palabras de cada conjunto

Como se puede observar en la Tabla 4.8, que presenta la proporción de palabras mal escritas en cada ejemplo con respecto al total de palabras, en algún caso la proporción de errores en los conjuntos de tipo competitivo es mayor que en la de no competitivo (*trump-2016* tiene el valor más alto y *amazon-4000* el valor más bajo). Sin embargo, vemos que algunos de los ejemplos del tipo competitivo tienen una proporción menor de palabras mal escritas que los de tipo reseña, por ejemplo *biden-2020* o *trump-2020* son mejores, bajo el punto de vista de la corrección de palabras, que los ejemplos de las reseñas para las aerolíneas. Por lo tanto, parece que la diferencia que subsiste tras eliminar de los textos los elementos de *Twitter* no se puede explicar simplemente por las palabras mal escritas, lo que refuerza la idea de que la sintaxis no juega un papel tan importante como la semántica y el contexto.

Como resumen final de los experimentos podemos decir que realmente los textos de las elecciones, en los que obtiene mejores resultados el Método Sencillo, no son más complejos ni tienen más erratas, pero sí incluyen más características típicas de *Twitter*. Además, estas características parecen aportar información relevante al clasificador humano que Watson no puede extraer, lo que explica parte de la ventaja del Método Sencillo. Por último, a pesar de eliminar estos elementos se sigue teniendo que en dos de los tres ejemplos el Método Sencillo sigue dando mejores valores para la métrica seleccionada que Watson, lo que significa que hay otra parte de la información que permanece y que da ventaja al clasificador humano y por tanto al Método Sencillo.

4.7.1. Ejemplos de dependencia contextual

Para terminar esta sección veamos a título ilustrativo algunos ejemplos en los que Watson clasifica incorrectamente y el Método Sencillo correctamente.

Ejemplo 4.7.1 *Dataset: trump-2020. Etiqueta real: -1, predicha por Watson: +1.*
Texto: @pastormarkburns @realdonaldtrump i agree they were "targeted". targeted by their own arrogance, ignorance and utter disregard for others. #rosegardenmassacre #rosegardensuperspreader #trumpincompetence #trumpmurdered208kamericans #trumpmustgo.
Comentario: Los hashtags indican claramente que el tweet *es negativo para D. Trump, y así lo detecta el método sencillo, seguramente porque alguno de los hashtags ya aparecía en el conjunto de entrenamiento. Sin embargo Watson lo marca como positivo, quizás porque el mensaje parece dirigirse hacia otras personas y no hacia el propio D. Trump.*

Ejemplo 4.7.2 *Dataset: trump-2020. Etiqueta real: +1, predicha por Watson: -1.*
Texto: @kamalaharris biden is going to jail with hillary and obama @realdonaldtrump make it so mr president!! god bless america.
Comentario: En este caso la dificultad no está en los hashtags u otros elementos sino en el contexto exterior: se habla mal de los contrincantes y respetuosamente de Trump, pero Watson no sabe quién es "biden" ni quién es "mr President" e infiere un valor negativo a partir del tono general negativo del mensaje.

Ejemplo 4.7.3 *Dataset: biden-2020. Etiqueta real: -1, predicha por Watson: +1.*
Texto: if they win this time they plan to never lose again, statehood for pr, get rid of electoral college, open borders for all new government dependent voters, massive programs for their "groups" pack scotus, etc etc.
Comentario: El mensaje hace eco de un rumor extendido por los seguidores de Trump de que si Biden ganaba sería el fin de la democracia en América. Seguramente a través del conjunto de entrenamiento el método sencillo ha "aprendido" que palabras como "electoral" y "college" o "borders" en relación con J. Biden suelen tener un significado negativo. Sin embargo Watson carece de ese contexto y marca el mensaje como positivo.

Ejemplo 4.7.4 *Dataset: biden-2020. Etiqueta real: -1, predicha por Watson: +1.*
Texto: @mariashriver @joebiden on the one hand you have a president who donates all of his salary, and on the other- a candidate who has enriched himself and his family with pay to play schemes that undermine our country. so yeah- they are polar opposites.
Comentario: Este mensaje es realmente difícil de clasificar para Watson por que se habla del "presidente" y del "candidato". El clasificador humano sabe que el presidente en ese momento era Trump y el candidato Biden, pero para Watson es difícil deducir tal información.

Capítulo 5

Conclusiones y Trabajo Futuro

Este capítulo cuenta únicamente con dos apartados. El primero de ellos expone las conclusiones finales obtenidas a través del desarrollo de este trabajo y el segundo plantea posibles trabajos futuros que pueden llevarse a cabo a partir de este.

5.1. Conclusiones

Los resultados del trabajo se pueden resumir en tres contribuciones:

1. En relación a la primera pregunta *P1: ¿Hay ejemplos reales en los que Watson obtenga peores resultados que un clasificador específico entrenado manualmente?* Hemos probado que la respuesta es afirmativa al encontrar ejemplos en los que un método sencillo basado en la clasificación manual puede superar a un sistema de tratamiento de lenguaje natural pre-entrenado de la magnitud de *IBM Watson*. Es importante enfatizar que el método utilizado ha sido muy simple y que tiene gran capacidad de mejora, pero justo ese era el objetivo. Por tanto los métodos de clasificación manual siguen teniendo sentido actualmente, al menos para ciertos casos. Además los ejemplos encontrados en los que esto sucede no son muy extraños ni rebuscados: se trata de *tweets* de elecciones estadounidenses, en inglés, el idioma en el que se supone que Watson muestra mayor potencia.

2. En relación a la segunda pregunta *P2: Si hay diferencias ¿podemos detectar a qué se deben? ¿tienen influencia aspectos como el uso de elementos utilizados comúnmente en Twitter (hashtags , URLs embebidas...), las palabras mal escritas o la complejidad lingüística del texto?* A través de los experimentos que hemos realizado para comprobar cada uno de estos factores que podrían explicar la diferencia de resultados entre ambas técnicas para los dos distintos tipos de texto podemos concluir que:

   - Para textos de tipo competitivo el uso de elementos específicos de *Twitter* es mucho más frecuente que para los textos de tipo reseña.
   - La complejidad lingüística del texto no es un factor que pueda provocar que el Método Sencillo logre mejores resultados para los textos de tipo competitivo, ya que como se ha podido observar en el apartado 4.5 al eliminar los elementos de *Twitter* la complejidad queda similar para ambos tipos de texto.
   - Además, la repetición de los experimentos pero eliminando los elementos específicos de *Twitter* realizados en el apartado 4.6.2 nos indican que estos elementos

tienen relevancia en los resultados del Método Sencillo, sobre todo en los textos de tipo competitivo (donde son más frecuentes), ya que el valor de κ decrece significativamente. Sin embargo, en todos los casos excepto uno se sigue cumpliendo que el Método Sencillo es mejor para los textos de tipo competitivo y Watson para los tipo reseña, por lo que los elementos específicos de *Twitter* explican parte de la superioridad del Método Sencillo en los textos competitivos, pero no toda.

- Por otra parte queda descartada la hipótesis de la diferencia por causa sintáctica ya que como se ha podido observar en el apartado 4.7, en primer lugar, la proporción de palabras mal escritas en general es despreciable y, en segundo lugar, no se da en mayor proporción en un tipo de texto u otro.

3. Por último, como contribución adicional se ha definido una nueva métrica basada en la Kappa de Cohen a la que hemos llamado Kappa Penalizada, que nos ha mostrado mediante los experimentos su buen funcionamiento y su superioridad de precisión con respecto a la Kappa original, que no tiene en cuenta la penalización por distancia, y con respecto a la Kappa con Pesos, que a pesar de contar también con una forma de penalizar el alejamiento de la clasificación real y la predicha, como se ha podido observar tiene comportamientos anómalos. Por lo tanto, se puede concluir que esta métrica es útil e interesante de estudiar en futuros trabajos.

5.2. Trabajo futuro

El trabajo futuro se puede dividir en tres líneas.

5.2.1. Estudio de $p\kappa$

Aunque en el apartado 3.5.2 se ha mostrado experimentalmente que $p\kappa < \kappa$ para valores de κ suficientemente altos sería interesante profundizar en la relación entre ambas métricas. En particular sería interesante comprobar si existe alguna cota α tal que para toda matriz de confusión con $\kappa > \alpha$ se tenga $p\kappa < \kappa$. Además para los casos de κ pequeños o negativos, en los que sí se encuentran valores de Kappa Penalizada que son mayores que Kappa, sería de interés determinar qué características tiene que tener una matriz de confusión para que suceda esto. Finalmente, esta definición de Kappa Penalizada abre el camino a generalizaciones donde la penalización no sea simplemente la distancia numérica entre posiciones sino otros valores fijados por el usuario.

5.2.2. Nuevos experimentos

Aunque hemos probado experimentalmente que Watson funciona mejor con los datos de las reseñas y peor en el contexto de las campañas electorales, un posible trabajo futuro sería realizar esta comparación con otros conjuntos de datos, incluyendo, por ejemplo, otros eventos competitivos como partidos de fútbol, etc. También sería interesante estudiar la dependencia del lenguaje, para ver si Watson se comporta mejor en inglés que en otros idiomas de los que admite. Otra línea de experimentación sería aplicar análisis de sentimiento a otras redes sociales, *Instagram*, *Reddit* (interesante por tratarse de textos más largos), comentarios de *YouTube*, y comparar las diferencias entre Watson y el Método Sencillo en relación con la red social concreta y sus características.

5.2.3. Profundizar en las razones por las que el Método Sencillo puede ser mejor que Watson

Ya hemos encontrado que esto se explica parcialmente por el contenido semántico de elementos como *hashtags* o el contenido de *URLs*. Falta por entender el valor de otros símbolos como los emoticonos. Además, hemos visto que estos elementos no explican totalmente la diferencia, lo que parece sugerir que hay palabras que cambian de definición en el contexto de la campaña electoral. Sería deseable diseñar experimentos, quizás con la ayuda de herramientas de tratamiento de lenguaje natural que permitan determinar qué expresiones concretas confunden a Watson.

Chapter 6

Introduction

In this introductory chapter we discuss in the first place the motivation that has carried us to the development of this work and we describe the objectives that this work attempts to solve. Then we present the term "sentiment analysis" and some of the automatic tools available. Eventually, we will describe the structure that this document follows.

6.1. Motivation

In recent years the use of automatic or semiautomatic tools employed to accomplish machine learning tasks such as sentiment analysis has been spread. That is the case with *IBM Waston* (Hig12), the IBM enterprise system that, among many other features, includes sentiment analysis. The main advantage of these systems is that we put aside all the work related with the elaboration of a machine learning system, especially the tedious task of classification by hand in case of supervised learning. The user of the system simply sends the text that is wanted to be analyzed and the target about which he desires to discover the transmitted opinion, and then the system returns this opinion as negative, positive or neutral. This system is founded on previously trained models drawn from a great amount of texts, this should also imply that we obtain better results than the specific models trained with a few examples.

However, the doubt about whether these systems in general overcome the specific models or not persists; there are cases where manual training is still required in order to obtain better results. In particular, in this work we envisage whether, in the case of messages generated by users in social networks related with competitive events such as electoral processes, the Watson previously trained models could perform worse than in the other cases. This is because, a priori, it seems that this type of messages have a strong contextual dependence on external news and also that given a large amount of messages and the intensity of news it leads to developing his own specific language where some words may have a different meaning than usual, whereas Watson assumes this usual meaning.

6.2. Objetives

The main aim of this project is to find an answer to the next two questions:

- Q1: Are there real examples in which Watson obtains worse results than a specific manually trained classifier?
- Q2: If there are differences, can we detect what is the reason? Do aspects such as the use of elements commonly used on *Twitter* (*hashtags*, embedded links...), misspelled words or the linguistic complexity of the text have influence?

Regarding the question Q1 we know that Watson can have poor performance if, for example, the language of the input text is not the same that the one that we have indicated to the system, namely, if we indicate that the language text is English and in fact it is Spanish. Obviously in this case we are tricking the system and it is not what we intend, but this suggests that something similar could happen in an English text if the words keep distance from the standard English language. This "distance" can be of two types:

1. Syntactic, because of misspelled words or unknown symbols to Watson.
2. Semantic, where words are apparently right but they have an unusual meaning, for example because the reader relates some keywords to external news. If it happens, the human trainer will take into account this context whereas Watson perform worse because it just take into account the standard meaning. Watson has another problem referring to the context: there are more than one target in competitive type texts, and frequently the message alludes indirectly to them (see Example 4.7.2) so it is easy that Watson does not infer the sentiment because of the difficulty of knowing who is the subject, while in review type texts there is just a single target.

Notice that this two options are not necessarily excluding, it may be that there are symbols in the text that Watson cannot understand (sintactic problem) but that the person who classifies the text does understand the meaning inside the context where the message was delivered; such as the use of *hashtags* in *tweets*, whose relevance, especially in political issues, is well-known.

Focusing in the semantic or context dependence, in order to prove whether Watson behaves differently we are going to use two types of texts, the first ones based on opinions about products, where we trust that the meaning of the words are the expected, and the other ones: *tweets* generated during electoral processes where the context, the news that happen during this process, may affect the meaning.

It must also be stressed that we do not necessarily seek that Watson performs worse than the simple manual classifier, that we called "Simple Method", it would be enough if the difference between both types of examples vary significantly.

Answering the second question, Q2, if the answer of the first the question is affirmative so that there are text examples in which Watson and the Simple Method obtain significant differences, we will need to explain what is the reason. It will be required to analyze several factors that could have influence, such as the elements frequently used in *Twitter*, and that could provide context that is included in external links or in the popular *hashtags*. On the other hand, it is important too to analyze whether other features of the texts such as the misspelled words that pre-trained systems like Watson cannot understand, as well as what happens with the introduction of an unknown language, have influence; another relevant factor could be the complexity of the text, which could affect Watson because of the usage of more or less complex texts.

This is needed in order to understand whether the type of context is the only cause that carries this supposed difference of the results between a simple machine learning method and Watson service or not. In this project we will distinguish two types of context: competitive context which occurs when it is referred to a limited time event such as football matches or electoral processes, and review context which does not refer to any concrete event but provides an opinion about a product or a service. It would be interesting to understand and analyze if these types of context are the only ones that explain the result differences or if, instead, the previously described factors also contribute, and the extent of each one.

6.3. Sentiment analysis

In contrast with the classic data mining techniques, when we talk about the sentiment analysis we do not purport to obtain *fact*, but *opinions* (L^+). Thus, this technique is also known as "opinion mining" (DLP03), although some authors differentiate both concepts. An informal definition but enough for our purpose could be: *process of extracting human thoughts and perceptions from unstructured texts* (HS19).

The set of techniques that allows the extraction of thoughts and perceptions constitutes a machine learning branch, but can also be considered as a part of data mining, natural language processing, computational linguistics, or even of sociology and psychology.

In particular, and sticking to machine learning, supervised machine learning techniques have been usually used in order to obtain the sentiment on a set of texts. The usual procedure is the following:

1. First of all, we select a subset of the texts traditionally called "training set".
2. An expert, mostly a human with knowledge about a specific subject, labels these texts drawing on the opinion that they state about a given topic. This labeling is often limited by: negative values (represented by -1), neutral (0) or positive (+1).
3. These texts along with their labels are employed with the aim of generating a model using some of the usual machine learning techniques. A previous step commonly applied is the *vectorization* of the texts, transforming them into numeric vectors.
4. Eventually, the model is employed so as to achieve the sentiment of the rest of the texts.

This usual procedure is long and burdensome, so many strategies that use previously trained models have come out. Although we will focus on the usage of *IBM Watson* system as a pre-trained method to sentiment analysis, there are many others tools that provide a similar functionality, one of the most renowned and easy to use is the *Python* library *TextBlob* (Lor) that provide several text processing functionalities, including sentiment analysis. This functionality assigns to a given text a polarity as well as a subjectivity score (AD17). The polarity is a value in the range [-1, 1] and the subjectivity score is a value in the range [0,1], where 0 indicates that the text is very objective and 1 indicates that it is very subjective.

6.4. Document structure

The structure of this document is divided into five chapters. The first one is the introduction, in the second one different development methodologies of data mining projects

are analyzed and the chosen methodology CRISP-DM is described along with its application in this project. Subsequently there is a chapter dedicated to the study of the usual evaluation metrics in works of this kind, in addition we define a new metric derived from *Cohen's Kappa* named Penalized Kappa that will be used in the next chapter: experiments chapter, where we present and evaluate our results trying to answer the questions that were asked in the objectives section included in this chapter. At last there is a chapter including the final conclusions and the future work of this project.

Furthermore, we have two last additional chapters. The first is this one, which corresponds to the English translation of the introduction that appears in the first chapter, and the second one which is also the English translation of the last main chapter: conclusions and future work.

The *Python* code developed for this project is located at `https://github.com/raqblanc/analsis-sentimiento-contexto-competitivo/`.

Chapter 7

Conclusions and Future Work

In this chapter there are only two sections. The first one presents the final conclusions obtained through the development of this project and in the second one we propose some possible future works that could be performed based on this one.

7.1. Conclusions

The results of this project could be summarized in three contributions:

1. Regarding the first question *Q1: Are there real examples in which Watson obtains worse results than a specific manually trained classifier?* We have proved that the answer is affirmative as we have found examples where a simple method based on manual classification can outperform a pre-trained natural language system as good as *IBM Watson*. We could highlight that the method we have used is very simple and that it has a great capacity to improve, but that was the point. Thus, manual classification methods are still relevant at least in some cases. In addition, the encountered examples in which this situation happens are not very strange nor cherry picked: they are *tweets* related to two American electoral processes, in English, the language in which Watson is supposed to be more powerful.

2. Concerning the second question *Q2: If there are differences, can we detect what is the reason? Do aspects such as the use of elements commonly used on* Twitter *(*hashtags*, embedded links...), misspelled words or the linguistic complexity of the text have influence?* Throughout the experiments we made in order to prove each of this factors that could explain the difference of the results between both techniques for the two different types of text we conclude that:

   - For competitive type texts the usage of *Twitter* specific elements is much more frequent than for the review type texts.
   - The text linguistic complexity is not a factor that may cause that the Simple Method gets better results for the competitive type texts, so as we have observed in section 4.5 when we remove the *Twitter* elements the complexity stays similar for both types of text.
   - Moreover, the replication of the experiments but removing the *Twitter* specific elements made in section 4.6.2 indicate that these elements have relevance in the Simple Method results, mainly in competitive type texts (where they are

more frequent), since the κ value decrease significatively. However, in all of the cases excepting one it is still true that the Simple Method is better for the competitive type texts and Watson for the other ones, so this *Twitter* specific elements explain part of the superiority of the Simple Method in competitive type texts, but not the whole of it.

- On the other hand, we put aside the hypothesis about a syntactic cause provoking the difference between the methods. As we observed in section 4.7, first of all the proportion of misspelled words is in general negligible and, in second place, this proportion is not greater in one type of text in relation with the other one.

3. At last, and as an additional contribution, we have defined a new metric based on Cohen's Kappa named Penalized Kappa. Throughout the experiments we have seen that this new metric has a great behaviour and it has a superior precision with regard to the original Kappa, that does not take into account the penalization because of the "distance", and with regard to Weighted Kappa, that despite the fact that it include a way to penalized this "distance" between the real and the predicted classification, as we have noticed it has an anomalous behaviour. Thus, we conclude that this metric is useful and interesting to study in future works.

7.2. Future work

The future work could be splitted into three lines.

7.2.1. Study of $p\kappa$

Even though in section 3.5.2 we have experimentally demonstrated that $p\kappa < \kappa$ for values of κ higher enough it would be interesting to dive into the relation between both metrics. In particular it would be interesting to prove that there is a peak value α so that for any confusion matrix with $\kappa > \alpha$ we have $p\kappa < \kappa$. In addition when κ is small or negative and $p\kappa > \kappa$ it would be interesting to determine which features have to have a confusion matrix so that this anomalous situation occurs. Eventually, this definition of Penalized Kappa allows a way out to generalizations where the penalization is not only the numeric distance between positions but other values setted by the user.

7.2.2. New experiments

Even if we have experimentally proved that Watson performs better with review type of texts and worse with a electoral processes context, a possible future work may be the realisation of a comparison between others datasets, including, for example, other competitive events such as football matches. It would be also interesting to study the dependence of the language in order to see if Watson has a better performance in English than in any other languages. Another line of experimentation could be applying sentiment analysis to other social networks, *Instagram*, *Reddit* (interesting because of the large size of texts), *YouTube* comments, and comparing the differences between Watson and the Simple Method in relation with the concrete social network and their differences.

7.2.3. Delve into the reasons why the Simple Method may be better than Watson

We have found that this is partially explained by the semantic context of elements such as hashtags or external links. We fail to understand the value of other symbols like emoticons. Besides, we have seen that these elements do not explain completely the difference, which could suggest that there are words that change their definition in the context of an electoral process. It would be desirable to design experiments, maybe using natural language processing tools that allow us to determine which concrete expressions confuse Watson.

Bibliografía

[AD17] Shreya Ahuja and Gaurav Dubey. Clustering and sentiment analysis on twitter data. In *2017 2nd International Conference on Telecommunication and Networks (TEL-NET)*, pages 1–5. IEEE, 2017.

[Agg18] Charu C Aggarwal. *Machine learning for text*. Springer, 2018.

[AS08] Ana Isabel Rojão Lourenço Azevedo and Manuel Filipe Santos. Kdd, semma and crisp-dm: a parallel overview. *IADS-DM*, 2008.

[ASNH17] Ika Alfina, Dinda Sigmawaty, Fitriasari Nurhidayati, and Achmad Nizar Hidayanto. Utilizing hashtags for sentiment analysis of tweets in the political domain. In *Proceedings of the 9th International Conference on Machine Learning and Computing*, pages 43–47, 2017.

[Ban] Shivam Bansal. Librería textstat. URL.

[Bar] Tyler Barrus. Librería pyspellchecker. URL.

[Coh60] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[DC48] Edgar Dale and Jeanne S Chall. A formula for predicting readability: Instructions. *Educational research bulletin*, pages 37–54, 1948.

[DLP03] Kushal Dave, Steve Lawrence, and David M Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web*, pages 519–528, 2003.

[Eur] Smart Vision Europe. What is the CRISP-DM methodology? URL.

[Fer12] David A Ferrucci. Introduction to "this is watson". *IBM Journal of Research and Development*, 56(3.4):1–1, 2012.

[FH02] Andy Field and Graham Hole. *How to design and report experiments*. Sage, 2002.

[Fle14] Rudolf Flesch. How to write plain english: A book for la wyers and consumers. 2014.

[Hig12] Rob High. The era of cognitive systems: An inside look at ibm watson and how it works. *IBM Corporation, Redbooks*, 1:16, 2012.

[Ho95] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[HS19] Fatemeh Hemmatian and Mohammad Karim Sohrabi. A survey on classification techniques for opinion mining and sentiment analysis. *Artificial Intelligence Review*, 52(3):1495–1545, 2019.

[Jen12] Kenneth Jensen. CRISP-DM process diagram. *No changes made*, 2012.

[Jha] Satyajeet Kumar Jha. kaggle-Twitter-US-Airline-Sentiment-. URL.

[L+] Bing Liu et al. Sentiment analysis and subjectivity.

[LDS+13] Mr Kamlesh Lahre, Mr Tarun Dhar, Diwan Suresh, Kumar Kashyap, and Pooja Agrawal. Analyze different approaches for ids using kdd 99 data set. *International Journal on Recent and Innovation Trends in Computing and Communication*, 1(8):645–651, 2013.

[Lor] Steven Loria. Librería textblob. URL.

[McA] Julian McAuley. Amazon product data. URL.

[Ng17] Andrew Ng. Machine learning yearning. URL, 2017.

[Pia] Gregory Piatetsky. CRISP-DM, still the top methodology for analytics, data mining, or data science projects (publicado en KDnuggets).
URL.

[Pow15] David MW Powers. What the F-measure doesn't measure: Features, Flaws, Fallacies and Fixes. *arXiv preprint arXiv:1503.06410*, 2015.

[Pow20] David MW Powers. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*, 2020.

[Qua] Quantum. Data Science project management methodologies.
URL.

[SBA+21] Noam Slonim, Yonatan Bilu, Carlos Alzate, Roy Bar-Haim, Ben Bogin, Francesca Bonin, Leshem Choshen, Edo Cohen-Karlik, Lena Dankin, Lilach Edelstein, et al. An autonomous debating system. *Nature*, 591(7850):379–384, 2021.

[TSD10] Efraim Turban, Ramesh Sharda, and Dursun Delen. Decision support and business intelligence systems (required). 2010.

[WH00] Rüdiger Wirth and Jochen Hipp. CRISP-DM: Towards a standard process model for data mining. In *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, volume 1. Springer-Verlag London, UK, 2000.

[Wue05] Karl L Wuensch. What is a likert scale? and how do you pronounce'likert?'. *East Carolina University*, 4, 2005.