ESTUDIO DE MEJORAS A LA ESTIMACIÓN DE PREFERENCIAS DE USUARIO EN SISTEMAS DE VALORACIÓN ONLINE

Trabajo Fin de Máster

Universidad Nacional de Educación a Distancia (UNED)
Escuela Técnica Superior de Ingeniería Informática
Máster Universitario en Investigación en Inteligencia Artificial
Convocatoria septiembre - 2021

Autor: Manuel Konomi Pilkati
Directores: Víctor Fresno Fernández y Roberto Centeno Sánchez

Contenido

Resumen. Este trabajo se centra en intentar encontrar un método de mejorar de la precisión en mecanismos de estimación de preferencias para usuarios de sistemas de reseñas online (ORS por sus siglas en inglés) utilizando exclusivamente los textos de los comentarios que dejan los usuarios en las propias reseñas. Para ello se intenta particularizar la elección de mecanismo de análisis de polaridad (MPC) que determina la estimación de preferencias de cada usuario en tiempo de ejecución. La elección del mecanismo MPC utilizado con cada usuario del sistema se realiza en base a información adicional extraída de los comentarios mediante técnicas de modelamiento de textos.

Palabras clave. NLP, modelado de textos, MPC, Machine learning, Clasificación.

1. Introducción.

En las últimas décadas, el crecimiento de internet y un mayor público ha hecho que, para las existentes plataformas de comercio electrónico, los comentarios y valoraciones de usuarios sean una parte fundamental de sus modelos de negocio (Lackermair, 2013) (Duan, Gu, & Whinston, 2008). Debido a esto, hay un esfuerzo constante por facilitar a los usuarios de este tipo de plataformas herramientas para expresarse. De este modo empiezan a surgir los *sistemas de valoración online* (*ORS* por sus siglas en inglés).

Normalmente, en la mayoría de estos sistemas, los usuarios deben mapear sus preferencias y opiniones internas hacia el objeto a una escala numérica (*rating)*. Así mismo, estas valoraciones suelen ir acompañadas de un comentario de texto opcional en el que el usuario puede añadir detalles concretos sobre la valoración no reflejados en el valor numérico.

El objetivo de estos sistemas (ORSs) es facilitar a los usuarios finales la búsqueda de nuevos objetos en base a las valoraciones de otros usuarios (creando un fenómeno conocido en la literatura como *Electronic Word-of-mouth*, o "boca a boca electrónico" (Thorsten Hennig-Thurau, 2004)). Esto convierte a los ORSs en una herramienta muy útil tanto para los usuarios, capaces de utilizar las opiniones de otros usuarios para tomar decisiones, o de afectar el éxito de un producto a través de sus valoraciones; como para las plataformas, capaces de extraer nueva información de su público para optimizar sus productos (Thorsten Hennig-Thurau, 2004).

Debido al crecimiento de estas plataformas de comercio electrónico, surge interés por extraer las *preferencias de los usuarios* de estas valoraciones.

Estas "preferencias de usuario" vienen dadas por el orden que toman los objetos del ORS (e.g. productos de Amazon, películas en IMDb, comentarios en AirBnb, reseñas de Google, etc.) que han sido valorados por el usuario. En otras palabras, las preferencias se podrían definir de una forma llana como gustos del usuario.

Estas preferencias son útiles de cara al desarrollo de nuevos productos y/o servicios que estas plataformas puedan lanzar en el futuro (Cui, 2012). Todo esto se traduce en diversos esfuerzos por capturar estas preferencias de, ya sea basándose en los ratings directamente o los comentarios (Ganu, 2009) (Leung, 2011).

No obstante, resolver la tarea de extraer preferencias de valoraciones apoyándose en ratings numéricos conlleva varios problemas (R. Centeno et al., 2018):

- En primer lugar, mapear una opinión a una escala no es siempre sencillo para un usuario (Sen., 2011). En el proceso puede involucrar muchos matices y sutilezas difíciles de plasmar en una escala numérica muy limitada.

- Un segundo problema sería el sesgo en las opiniones de los usuarios (Jøsang, 2009). Esto se refiere a que un usuario puede ser por lo general optimista en todas sus reseñas y otro usuario pesimista en las suyas, dando lugar a situaciones en las que el mismo objeto en la misma situación tiene valoraciones muy distintas.

- Finalmente, otro problema es el sesgo de selección (*selection bias*) (Dalvi, Kumar, & Pang, 2013). Este fenómeno se refiere a la tendencia a que las valoraciones, por lo general, se desvían hacia valores más positivos. Esto se debe según ellos a que un usuario suele estar expuesto a objetos altamente valorados, con una gran probabilidad de que la nueva valoración sea positiva.

Debido a estos problemas, los enfoques de estimación o extracción de preferencias de usuario basados en ratings pueden tener problemas ya que es posible que la representación final de las preferencias también esté sesgada.

Con la motivación de solucionar los problemas de sesgo de los ratings numéricos, surge la tarea de clasificación de polaridad multiclase (*Multiway Polarity classification-MPL*). MPL se trata de una versión más avanzada de la clasificación de polaridad simple, en la que se analiza el sentimiento de un texto de forma binaria (un texto es o bien positivo o negativo en su sentimiento). La diferencia de MLP está en que es capaz de analizar la polaridad/sentimiento en un rango más amplio (Seroussi, Zukerman, & Bohnert, A user-based approach to multi-way polarity classification., 2010).

Tratando de aprovechar el análisis de polaridad, aparece una aproximación que propone extraer las preferencias de usuario a base de mapear los comentarios que normalmente existen en las valoraciones en el rating numérico, y así usar este rating numérico para obtener las preferencias (Hermoso, Centeno, & Fasli, 2014).

No obstante, el problema de este enfoque es el mismo que el de los enfoques basados en el rating numérico del usuario, ya que el mapeo de la polaridad en una escala numérica puede conllevar perdidas en precisión a la hora de representar las preferencias debido a las limitaciones de la escala del rating (R. Centeno et al., 2018).

Por otro lado, con la idea de dar respuesta al primer problema (i.e. la capacidad de representación limitada de una escala de rating numérico), se han propuesto dos aproximaciones. La primera en la que se extraen las preferencias de un usuario a base de preguntar al usuario que compare objetos por pares de forma proactiva (Centeno, Hermoso, & Fasli, 2015). De este modo, el sesgo introducido a la hora de mapear la opinión de un usuario a un rating desaparece, ya que lo que se le pide es que decida el orden de todos los objetos valorados él mismo.

Sin embargo, este enfoque tiene el problema de necesitar una fuerte participación del usuario para funcionar, y se pierde la ventaja de analizar simplemente las valoraciones existentes en los ORSs de forma pasiva.

Con la idea de resolver los inconvenientes de los enfoques anteriores, el trabajo sobre el que nos basamos (R. Centeno et al., 2018) propone la extracción de preferencias de usuario a través de lo que denominan Rankings de Reputación Individuales (*IRR* por sus siglas en inglés).

Estos rankings comprenden un subconjunto de objetos del ORS valorados por un usuario en concreto que son ordenados en base a la preferencia de este por medio de la comparación a pares de nuevo.

La diferencia fundamental con aproximaciones anteriores es que, para llevar a cabo estas comparaciones por pares, se utiliza información extraída de los comentarios de texto de las propias valoraciones del usuario de forma directa. Es decir, sin mapearla en ninguna escala de rating numérico. Esta información viene en forma de un valor de sentimiento/polaridad del texto extraído a través de diversas familias de métodos de análisis de sentimiento. Este paso permite ignorar directamente los ratings numéricos para centrarse en dicho valor. Se consiguen así, en teoría, dos cosas. En primer lugar, se evita el problema de tener que evaluar un objeto en base a ratings sencillos (e. g. escala del 1 al 10) en favor de un valor de sentimiento con más

rango y, por otro lado, se evitan problemas de sesgo y de mapeo de una opinión a un rating numérico (R. Centeno et al., 2018).

Sin embargo, en los resultados del trabajo previo se puede observar que no existe ninguna familia de métodos de análisis del sentimiento que obtenga el mejor IRR para todos los usuarios de su experimento. En otras palabras, se plantea un escenario donde el usuario A consigue su mejor IRR con un método de análisis del sentimiento distinto al del usuario B.

De esta observación nace la motivación original de nuestro trabajo. Y podríamos resumirla de la siguiente manera: Si somos capaces de identificar que usuario genera su mejor IRR con que método de análisis del sentimiento, podríamos utilizar este cocimiento para elegir el método de firma dinámica durante la generación de IRRs, lo que conllevaría una mejora de las estimaciones de preferencias de usuarios de manera global en el sistema.

En este trabajo trataremos de responder a esta cuestión con el objetivo de refinar el proceso de estimación de preferencias y así mismo, intentar obtener alguna pista en que hace que un usuario tenga mayor afinidad con un método de análisis de sentimiento concreto. Y para ello se utilizará exclusivamente el comentario de texto de las reviews, que es la única información de la que dispondríamos en un caso de uso real.

El resto de este documento está estructurado de la siguiente manera: en la sección 2 se profundizará en el problema que acabamos de mencionar, así como la formalización de nuestra hipótesis y objetivos. En la sección 3 se exploran las tecnologías y recursos a nuestro alcance para llevar a cabo el trabajo (e.g. información que podemos extraer de un texto, formas de clasificar distintos usuarios, etc.). La sección 4 es una presentación del trabajo previo sobre el que nos basamos (R. Centeno et al., 2018). La sección 5 engloba una descripción detallada de nuestra propuesta. En la sección 6 se definen los datasets de los que partimos, así como una descripción de todos los parámetros de la solución implementada para el trabajo con los que experimentaremos. La sección 7 comprende la presentación de los experimentos y análisis de sus resultados . Finalmente, en la sección 8 se presentan nuestras conclusiones y posibles nuevos trabajos a realizar a la vista de los resultados obtenidos.

2. Descripción del problema, hipótesis y objetivos.

Una vez tenemos una idea general del contexto de este trabajo, en esta sección nuestra intención es hacer dos cosas. En primer lugar, definiremos el problema concreto que se intenta resolver (i.e. la estimación de preferencias mediante MPC) y, segundo lugar presentaremos nuestra hipótesis y objetivos para alcanzar la misma.

2.1. Descripción del problema.

En este trabajo el problema que se trata de resolver es el de estimar las preferencias de usuario en un OBS y valiéndonos de métodos de análisis de sentimiento (o MPC) para inferir dichas preferencias.

Para formalizarlo un poco, definiremos primero un ORS como un conjunto de usuarios {*U*} que participan en el sistema, un conjunto de entidades u objetos {*E*} y un conjunto de valoraciones o reviews {*R*}.

De esta forma, la review Rij existente en *R* es la valoración realizada por el usuario *Ui* sobre el elemento Ej dentro del ORS. Para este trabajo vamos a suponer, como se propone el trabajo que usamos de punto de partida (R. Centeno et al., 2018), que las preferencias de usuario

tienen un solo máximo (Black, 1948) (Conitzer, 2007). Dicho en otras palabras, las preferencias de un usuario no pueden empatar. Un usuario siempre preferirá entre objeto u otro.

Debido a este supuesto, existe una función *F* para el usuario *Ui* tal que *Fi* (*Ei*) -> *R* (real) siendo *Ei* un subconjunto de objetos en *E* sobre los que el usuario *Ui* tiene una preferencia interna. De esta manera, podemos definir la preferencia interna del usuario como la función *Fi* aplicada sobre el conjunto de objetos *Ei* (i.e. *pref* = {*Ei*, *Fi*} Esta función *Fi* permite representar la preferencia de un usuario por un objeto *Ei* en el ámbito de los números reales.

Por otro lado, definamos una estimación de las preferencias del usuario (*prefi*) como *Oprefi* (*Ei*). Consideramos *Ɛ*(*Oprefi*) como el error entre la estimación de la preferencia y la preferencia real *prefi*.

Con todo esto en mente, nuestro problema se podría definir de la siguiente forma:

Dado un ORS = {*U*, *E*, *R*}, un conjunto {*prefi*} de preferencias con un solo máximo para cada usuario *Ui* existente en *U*, y un subconjunto *Ri* de reviews en las que un subconjunto *Ei* de objetos, pertenecientes a *prefi*, han sido evaluados por dichos usuarios (i.e. reviews *Ri* realizadas por los usuarios *Ui*).

Adicionalmente, definamos *M* como el conjunto de métodos de análisis de sentimiento en el sistema. Dado esto, On(*prefi*) como la estimación de la preferencia de usuario *prefi* utilizando el método de análisis de sentimiento *Mn*. Por otro lado, no existe ningún *Mn* tal que *Ɛ* (On(*prefi*)) < *Ɛ* (On'(*prefi*)) para todos los *Mn*' en *M* distintos de *Mn* (i.e. ningún método de análisis vale para todos los usuarios) (R. Centeno et al., 2018).

Con todo esto, nuestro problema puede definirse como la creación de un conjunto de estimaciones {On(*prefi*)} de las preferencias internas de usuario {*prefi*} a partir del conjunto de reviews *R* realizadas por dichos usuarios *Ui* con el método de análisis del sentimiento *Mn* (R. Centeno et al., 2018).

2.2. Hipótesis y objetivos.

Una vez tenemos una definición concreta del problema, en esta sección utiliza dicha definición para desglosar, paso por paso, las **hipótesis** que motivan nuestro trabajo.

Haciendo un resumen de la sección anterior, nuestro problema es cómo crear un mecanismo que nos permita, para cada usuario en el ORS, asignar el método de análisis de sentimiento que permita crear un IRR (o, en otras palabras, estimar su preferencia de usuario) de la forma más cercana posible a la preferencia real de cada usuario a partir de sus reviews. Para ello, la primera hipótesis que planteamos es la siguiente:

Hipótesis 1: Podemos encontrar usuarios Ui existentes en el ORS tales que puedan clasificarse dependiendo de que método de análisis Mn minimice la función de error Ɛ(On(prefi)).

Suponiendo que esta hipótesis inicial es cierta, la primera pregunta que puede venir a la cabeza es "¿Cómo?". Para poder clasificar a los usuarios según el método de análisis de sentimiento que minimice el error necesitamos datos sobre los que clasificar. Esto nos lleva a nuestra segunda hipótesis:

Hipótesis dos: Existe un conjunto de atributos Ai que permitan la clasificación del usuario Ui de acuerdo con la hipótesis 1.

Con esta segunda hipótesis se empieza a ver más clara la posible solución al problema de clasificación, pero aún nos queda la incógnita de donde encontrar esos datos. En el problema tal y como se ha definido, señalamos que estos atributos tienen que venir del conjunto de reviews de un usuario dado *Ui* dentro del ORS. Con esto en mente se presentan las dos siguientes hipótesis.

Hipótesis tres: Existe un subconjunto de atributos textuales Ai' contenido en Ai generado a partir del subconjunto de reviews Ri de los usuarios Ui.

Hipótesis cuatro: Existe un subconjunto de atributos de usuario Ai'' contenidos en Ai y relativos al usuario Ui y generados a partir del subconjunto de reviews Ri.

Estas dos últimas hipótesis proponen que (i) es posible encontrar características relativas al texto (propiedades del texto en sí) generado por los usuarios que permita clasificar, y (ii) es posible encontrar características del propio autor a partir del análisis de sus reviews.

valiéndonos de las hipótesis que acabamos de definir, podemos presentar una serie de **objetivos** a cumplir de cara a probar estas hipótesis. Empezando por la primera hipótesis, el primer objetivo que tenemos es:

Objetivo 1: dado un ORS y unos métodos de análisis del sentimiento M, y un dataset etiquetado con el método Mn que produzca menor error Ɛ, encontrar un método de análisis de los textos de un usuario tal que nos permita asignar un método Mn a un usuario Ui para generar su IRR.

El requisito adicional que tenemos se trata de la necesidad de un dataset de un ORS en el que los usuarios vengan etiquetados con las clases que queremos asignarles, correspondiendo estas clases con los métodos disponibles en el conjunto *M*. Esto nos lleva a nuestro segundo e inescapable objetivo:

Objetivo 2: Generar un dataset a partir de un ORS y el trabajo previo en el que nos basamos que contenga etiquetado para los usuarios.

Llegados a este punto, nuestros primeros dos objetivos servirían para contestar a la primera hipótesis a falta de elegir que datos se usan para clasificar. Esto nos lleva al tercer y cuarto objetivo, que dicen así:

Objetivo 3: Dadas las reviews de un usuario Ui, encontrar e implementar un algoritmo o mecanismo que permita la extracción de atributos que contengan información referente al texto de las reviews.

Objetivo 4: Dadas las reviews de un usuario Ui, encontrar e implementar un algoritmo o mecanismo que permita la extracción de atributos que contengan información referente al autor de las reviews.

Estos dos objetivos contestan a las hipótesis 3 y 4 respectivamente. Y estas dos hipótesis, en su conjunto, conforman la hipótesis 2.

En la próxima sección se visitan distintas tecnologías disponibles en el campo de la inteligencia artificial y del procesamiento del lenguaje natural (NLP) para encontrar enfoques candidatos para resolver nuestro problema.

3. Estado del arte.

Antes de entrar a la propuesta formal de este trabajo, en esta sección vamos a hacer dar un vistazo general a los sistemas, tecnologías y metodologías que hemos estado comentando por encima en las secciones anteriores.

En el contexto del problema que se intenta resolver, nuestro punto de partida son los sistemas de valoraciones online, o por sus siglas en ingles ORSs (Fang, Zhang, J., & Zhu, 2013) (Mudambi, M., & Schuff., 2010). Estos sistemas contienen colecciones de entidades que van a valorarse por la razón que sea. Por un lado, tenemos ejemplos como Amazon, donde la lista de objetos que existe en el sistema son los productos a la venta. Por otro lado, también tenemos ejemplos como Internet Movie Database (IMBd), en la que la colección de entidades es fichas de series y películas. Estos sistemas permiten que un usuario registrado en el mismo pueda seleccionar una entidad y dejar una valoración. Por lo común, estas valoraciones son de carácter numérico en una escala, y suelen tener asociado un comentario de texto para que el usuario pueda expresarse con más libertad. Históricamente, en la tarea de estimación de la preferencia de usuario se ha confiado en los ratings numéricos ofrecidos por estos mismos (Beuscart, Mellet, & Trespeuch) (Luca, 2016), aunque otros trabajos indican los efectos de los comentarios de texto en las ventas para ORSs enfocados al comercio electrónico (Ren & Nickerson).

En este punto, tenemos ORSs con ratings numéricos, que como hemos mencionado tienen problemas de sesgo y de representación, y comentarios ricos en información pero que no se están utilizando. Como ya hemos comentado, (Sen., 2011) señala que el mero hecho de mapear una opinión en un rating numérico es un proceso complejo y vulnerable a distintos sesgos según el contexto de la valoración. Este dato nos hace partir de un valor inicial ya comprometido. Si a esto le sumamos los sesgos señalados por (Jøsang, 2009) y (Centeno, Hermoso, & Fasli, 2015), el valor del rating como mecanismo de extracción de preferencias puede ponerse en entredicho.

Frente a este panorama surgen propuestas en las que se sugiere que se puede obtener preferencias de usuario con mayor precisión haciendo que estos usuarios comparen pares de objetos en lugar de mapear una opinión abstracta en un rating numérico. De esta manera se obtendría un ranking con las preferencias de ese usuario más representativo.

Esto podemos verlo en trabajos como el de (Balakrishnan & Chopra, 2010) donde se utiliza esta idea para refinar el aprendizaje de sistemas de recomendaciones preguntándole a los usuarios si “prefieren A o B”. Otra versión de esta idea está también presente en (Hermoso, Centeno, & Fasli, 2014) donde se intenta mejorar la forma de crear rankings globales de objetos comparando rankings individuales de usuarios. Estos rankings estarían generados a base de comparación por pares de sus valoraciones. El objetivo de este enfoque es el de comparar las preferencias de los usuarios a la hora de crear un ranking global en lugar de agregar los ratings sin contexto alguno.

De esta forma, llegamos a situaciones en las que, si un usuario valora todos los objetos de un ORS con un sesgo muy negativo, esto no afectaría tanto al ranking global ya que lo que se tendría en cuenta es el orden (su preferencia) de sus valoraciones.

El problema principal de estas aproximaciones es que son intrusivas de cara al usuario. Si este decide no comparar múltiples parejas de objetos valorados, estos sistemas no sirven de mucho.

Un enfoque alternativo hace uso del hecho de que muchos de estos ORS tienen asociado al rating de la valoración un comentario de texto que puede ser explotado para mejorar la extracción de preferencias. Un ejemplo interesante de este tipo de enfoque es el que se presenta en (Seroussi, Zukerman, & Bohnert, Collaborative Inference of Sentiments from Texts., 2010). En este plantean el análisis de sentimiento en los textos del usuario, pero a la hora de extraer las preferencias, tienen en cuenta la similitud de un usuario con el resto de conjunto de usuarios en el ORS. Esto hace posible que los ratings inferidos por análisis de sentimiento sean ajustados en base a características del propio usuario y no las valoraciones en sí mismas.

A pesar de las ideas que traen estos distintos enfoques, se sigue dependiendo del mapeo de la preferencia de usuario a un rating numérico, que como hemos mencionado anteriormente, implica la posibilidad de transformar la preferencia de usuario y reducir su precisión. Así llegamos a nuestro punto de partida, donde (R. Centeno et al., 2018) proponen ignorar esos ratings numéricos y combinan la idea de los rankings individuales por comparaciones de parejas con la idea de representar la preferencia directamente con el análisis de sentimiento. Dando lugar a los IRRs.

Con la intención de iterar en el mecanismo de los IRRs y de resolver nuestra pregunta planteada en la sección 2 (i.e. ¿es posible escoger el mejor método de análisis de sentimiento de un grupo de métodos dado para estimar las preferencias del usuario basándonos en el corpus de sus comentarios de texto?) debemos explorar primero que tareas a resolver para contestar a esa pregunta y, a continuación, que herramientas hay a nuestra disposición para llevar a cabo dichas tareas.

Dentro del contexto que es la extracción de preferencias de usuario en un ORS mediante el uso de IRRs de (R. Centeno et al., 2018), nuestro problema puede enfocarse como un problema de clasificación en el campo de NLP. En este problema, U representa el conjunto de usuarios de un ORS, C representa el conjunto de métodos de análisis de sentimiento disponibles, siendo estos nuestras clases, de tal forma que un usuario u_n pertenece a una clase c_m siempre que el $IRR_{n,m}$ es mejor que cualquier $IRR_{n,x}$ donde c_x representa el resto de los métodos dentro de C. Entendemos por "mejor" que el IRR se acerque más al ranking teórico ideal del usuario u_n. Mas adelante en la sección 6 se explica cómo se calcula la precisión de estos IRRs.

Una incógnita en este problema de clasificación es la entrada. Partiendo de nuestra hipótesis, la entrada al clasificador tiene que estar relacionada de alguna forma con el corpus de comentarios de texto del usuario, sin embargo, esto puede conseguirse de muchas formas distintas (e.g. meter el valor binario de la cadena de texto como entrada, o la suma de las posiciones que ocupan las palabras utilizadas en un diccionario). Debido a esto, se separa el resto de este apartado en dos secciones. Por un lado, la exploración de métodos de clasificación, y, por otro lado, la caracterización del texto y la información que se puede extraer del mismo.

3.1. El problema de la clasificación.

En el ámbito de los clasificadores, en el campo del aprendizaje en IA (*Machine Learning*), (Kelleher, Mac Namee, & D'arcy, 2020) nos propone distinguir entre cuatro grandes familias que utilizaremos como punto de partida. Las familias son las siguientes: modelos de aprendizaje basados en información, modelos basados en similitud, modelos basados en probabilidad, y modelos basados en error.

Empezando por los modelos basados en información, (Kelleher, Mac Namee, & D'arcy, 2020) pone como ejemplo los árboles de decisión, donde el modelo puede cumplir la tarea de

clasificación navegando a través de una estructura de árbol en la que los atributos de entrada se utilizan decidir que ramas del árbol se van a seguir. Una ventaja de este tipo de algoritmos es que son transparentes en cuanto a cómo llegan a sus conclusiones, lo que permite explicar porque se obtienen los resultados que se obtiene. Sin embargo (Martens, Vanthienen, Verbeke, & & Baesens, 2011) señalan que a cambio tienen un peor rendimiento en comparación con otras familias.

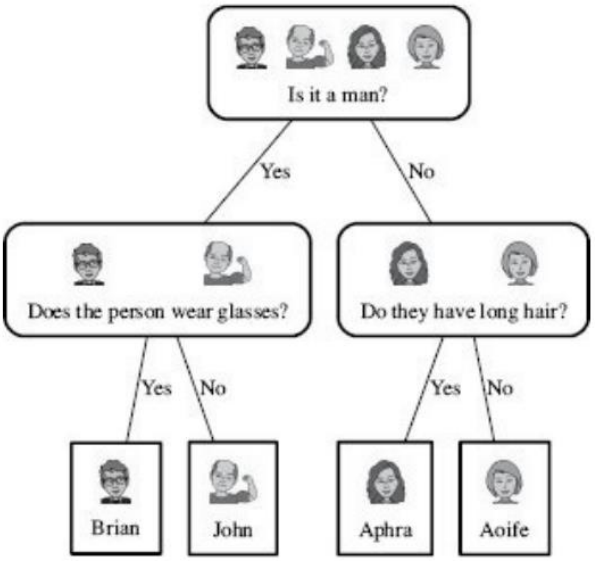


Ilustración 1: Ejemplo del funcionamiento de un árbol de decisión basado en el juego "Quién es quién?". Imagen tomada de (Kelleher, Mac Namee, & D'arcy, 2020)

La siguiente familia es la de modelos basados en similitud. Según (Kelleher, Mac Namee, & D'arcy, 2020) este tipo de modelos intentan clasificar un elemento basándose en objetos similares que ya conozcan. De este modo, si el modelo tiene en memoria un objeto o_1 con un vector de atributos v_1, y pertenece a una clase c_1, un nuevo objeto o_n desconocido pertenecerá a la clase c_1 si el vector del nuevo objeto se parece lo suficiente a v_1.

	Grrr!			Score
	✓	✗	✗	1
	✗	✓	✗	1
	✗	✓	✓	2

Ilustración 2: Ejemplo de tabla de decisión de un modelo basado en similitud. Las columnas corresponden a atributos descritos, y las filas a animales conocidos en la base de datos. Imagen tomada de (Kelleher, Mac Namee, & D'arcy, 2020). Originalmente de Jan Gillbank

En la ilustración 2 se puede ver un ejemplo práctico de esto. En esta tabla las filas representan objetos en nuestra base de conocimiento, y las columnas representan rasgos descritos de un nuevo objeto (animal) en este caso. El nuevo animal gruñe, tiene membranas interdigitales y pico. Con esa información, el modelo procede a ver cuántos criterios satisfacen los animales en nuestra base de conocimiento, y clasifica el animal (a pesar cumplir todos los criterios) como un pato por tener mayor semejanza. Como representante de este tipo de modelo nos propone KNN (K-Nearest-Neibourghs). Las ventajas de este tipo de modelo es que también es sencillo de explicar. Conociendo la razón por la que los objetos anteriores pertenecen a una clase, se puede razonar porque un nuevo objeto pertenece a la misma. En cuanto a desventajas, este modelo no va a darnos una formula general para la clasificación (como serían los modelos basados en reglas como el árbol de decisión), si no que contestará comparando exclusivamente con los datos que ya posee.

A continuación, tenemos los modelos basados en probabilidad. Este grupo de clasificadores (que engloba los clasificadores bayesianos y todos sus derivados) calculan la probabilidad de que un objeto pertenezca a una clase dados sus atributos y, durante su funcionamiento van ajustando estas probabilidades según reciban nueva información.

Finalmente, en la división que hemos utilizado, tenemos los modelos basados en error. Reciben este nombre ya que se trata de modelos que mejoran su precisión a base de entrenar con sets de datos en los que se conoce la clase de los objetos. Al intentar clasificar un objeto, el modelo se reajusta en base al error para mejorar su rendimiento (Kelleher, Mac Namee, & D'arcy, 2020) (Ansari, Erol, & & Sihn, 2018). Como ejemplos en este campo tenemos los diferentes modelos de regresión lineal, modelos de RandomForest, modelos de red neuronal artificial (ANN) y de SVN. Estos modelos tienen un gran rendimiento y precisión a la hora de predecir o clasificar datos, pero a cambio es prácticamente imposible discernir que lógica utilizan para llegar a las conclusiones que obtienen. En otras palabras, no tienen ninguna transparencia y trabajan como una caja negra. Esto es una desventaja a la hora de querer entender en mayor profundidad los resultados de un problema de clasificación.

3.2. Modelamiento de textos.

Una vez tenemos una idea de cómo podemos clasificar usuarios y que alternativas nos da el campo del *Machine Learning*, vamos a explorar las posibilidades a nuestro alcance para transformar los textos de usuario a algo que estos modelos puedan utilizar. Debido a que la tarea se refiere a obtener información, de alguna manera, y debido a que esto es un trabajo en el campo de la inteligencia artificial, nuestro campo de interés en esta sección es el del procesamiento del lenguaje natural (*NLP – Natural Language Processing*).

Dentro de este campo existe, hoy en día, un gran interés por innovar y desarrollar nuevos mecanismos para mejorar el tratamiento del lenguaje natural, especialmente por parte de grandes compañías tecnológicas (e.g. Google con BERT y DeepMind, IBM con Watson). Esto se debe a la cantidad de aplicaciones que se benefician del campo de NLP.

En (Khurana, Koli, Khatter, & Singh, 2017) nos dan varios ejemplos de esto, como es el caso de la **traducción** de idiomas, una aplicación que a pesar de no ser perfecta tiene un valor inmenso para romper barreras de idioma.

También encontramos la **clasificación de textos**. Esta aplicación hace posible la clasificación de colecciones de documento según su naturaleza (e.g. categorizar artículos

periodísticos en base a los temas que tratan). Dentro de la clasificación de textos podríamos destacar los **filtros de spam**, que se basan en esta tecnología.

Una aplicación más útil en nuestro contexto es la **extracción de entidades de un texto** también mencionada por (Khurana, Koli, Khatter, & Singh, 2017). Este tipo de extracción busca, en un texto de origen no preprocesado, entidades como compañías, personas, artículos, periodos de tiempo, cantidades y más, con el objetivo de facilitar el procesamiento de grandes volúmenes de documentos y la extracción de palabras clave. Estas palabras clave -o entidades- se utilizan para tener una idea del contenido del documento. Esta aplicación sería también útil para la clasificación de documentos vista anteriormente. En (Radhakrishnan, 2018), por ejemplo, utilizan este tipo de sistemas para la creación de *Knowledge Bases* (KBs) en las que almacenan todas las entidades de un corpus de documentos de forma estructurada para mejorar las consultas hechas por usuarios. Otro ejemplo es (Nguyen & al., 2017), donde utilizan su sistema de extracción de entidades nombradas para facilitar la lectura de una web de noticias japonesa a usuarios que estén aprendiendo japones, añadiendo anotaciones en las distintas entidades del texto. En (Nadeau & Sekine, 2007) exploran en profundidad las diferencias en el soporte de esta subcategoría según los idiomas de los textos, sus dominios de conocimiento, el género al que pertenezcan (e.g. ensayos, artículos, textos científicos, etc.) o las entidades a extraer.

Si seguimos avanzando en el campo de los sistemas NLP, tarde o temprano nos acabamos encontrando con los embeddings. Empezando por los **Word Embeddings** (Almeida & Xexéo, 2019) nos explican que los embeddings se tratan de una forma de representar el contenido de un texto. A diferencia de aplicaciones anteriores como la extracción de entidades, en este caso la representación del texto no es comprensible para un ser humano (Bakarov, 2018).

En los Word Embeddings (WE de aquí en adelante), las palabras de un idioma se representan en un vector de tamaño fijo N, de modo que cada palabra ocupa una posición en este espacio N-Dimensional. La distribución de las palabras en este espacio viene dada por la coocurrencia de estas (Turian, Ratinov, & Bengio, 2010). Dicho de otro modo, palabras que estadísticamente aparecen juntas a menudo en textos, estarán cerca en el espacio n-dimensional. Esta aproximación, a pesar de la desventaja de ser muy opaca para las personas y en ocasiones la falta de datasets de entrenamiento adecuados (Bakarov, 2018), tiene las ventajas de caracterizar de forma efectiva las relaciones semánticas entre palabras, permitiendo tareas como la predicción de la siguiente palabra dada una secuencia, así como de ser compatible con modelos de ML dada su forma de vector (Almeida & Xexéo, 2019). Otros ejemplos de WE los encontramos en (Fernando Enríquez, 2016), donde se compara el uso de WE con Word2Vec para la tarea de clasificación de documentos con el método Bag of Words. Otro ejemplo sería (Galke, 2017) en el cual utilizan WE para la tarea de consulta de información, en la que comparan los valores que se extraen de una consulta (como en un motor de búsqueda) con los valores obtenidos de los documentos en la base de datos.

Siguiendo la filosofía de los WE otro método de caracterización conocido como **Sentence Embeddings.** La idea a grandes rasgos es similar a la de WE. En este caso en lugar de distribuir palabras en un espacio n-dimensional, lo que se distribuye son textos.

Podemos encontrar varios ejemplos de esta subcategoría de NLP, como en (C. Zhang, 2017), que utiliza Sentence Embeddings para la tarea de creación de resúmenes de textos o para parafrasear los textos originales. Por otro lado, en (Duong, 2019), utilizan Sentence embeddings para la comparación de términos de Ontología Genética (términos que describen un gen y el

resultado directo de su expresión, ya sea ARN o proteínas) en una base de datos sin depender de la estructura de árbol en la que se almacenan estos términos para inferir su similitud. Finalmente, se podría profundizar en las distintas variedades de Sentence Embeddings y su eficacia con distintos tipos de documentos, repasando los trabajos de (Conneau, Kruszewski, Lample, Barrault, & Baroni, 2018) y (Krasnowska-Kieraś & Wróblewska, 2019).

Hasta este momento hemos revisado distintas formas de caracterizar textos, ya sea en entidades concretas, en vectores que representan palabras y vectores para representar oraciones. Estas formas de representación se pueden utilizar para la extracción de diversos tipos de información en los textos, ya sea su aplicación en motores de búsqueda, traducción de idiomas, o -de interés para nosotros- análisis de sentimiento (Khurana, Koli, Khatter, & Singh, 2017). A continuación, vamos a explorar algunas de estas aplicaciones del campo de NLP que podríamos definir como "extracción de información implícita".

Dentro de este bloque de aplicaciones, tenemos **análisis de polaridad**. Este tipo de análisis determina si el texto es positivo, neutro o negativo. Algunas versiones añaden categorías adicionales como "muy positivo" y "muy negativo", y otras pueden devolver un valor continuo dentro de un rango para determinar hacia donde tiende. Estos sistemas se suelen basar en lexicones (diccionarios con valores, de polaridad en este caso, asignados a cada palabra). Este tipo de análisis es muy útil a la hora de evaluar textos subjetivos, como opiniones o comentarios acerca de algún objeto o entidad. Como ejemplo de este tipo de extracción de información, tenemos (Pawar, 2017) donde se implementa un sistema para analizar la polaridad de opiniones en mensajes de Twitter. También podemos ver el uso de esta tecnología en (R. Centeno et al., 2018), donde la polaridad define el sentimiento extraído por SOCAL y SVR. Dependiendo del tipo de textos con los que se va a trabajar, las técnicas específicas que se utilicen o los lexicones elegidos, se pueden encontrar múltiples configuraciones de esta aplicación como ilustra (Hussein, 2018) en su comparativa.

Similar a los sistemas de análisis de polaridad, tenemos **detección de emociones**. Estos sistemas no se limitan a clasificar un texto como positivo o negativo, si no que evalúan varias categorías emocionales (e.g. alegría, tristeza, ira, frustración, etc.). Al igual que los de polaridad, pueden utilizar lexicones para obtener estas categorías, o pueden obtenerlas a través de modelos de machine learning. También tienen aplicaciones en la evaluación de textos de carácter subjetivo o de tipo personal. Un ejemplo de esto lo podemos ver en (Oh, Lee, Ko, & Choi, 2017). En su trabajo se ha implementado un chatbot que hace las veces de consejero o asesor psiquiátrico. Tal y como ellos comentan, en este problema es muy importante detectar las emociones que presenta el usuario para responder de forma adecuada a su estado emocional. Es interesante señalar la distinción mencionada en (Alswaidan & Menai, 2020) donde separan la detección de emociones en emociones explicitas (i.e. donde se detecta la emoción buscando palabras clave como "alegre" o "triste" en el texto) e implícitas (i.e. detección de una emoción a través de la relación entre palabras que en sí mismas no indican necesariamente emociones). Esta ultimo tipo en particular es especialmente difícil según (Alswaidan & Menai, 2020) ya que no basta con conocer las definiciones de las palabras, sino que también es necesario *entender* el significado que surge de sus relaciones. Esto tocaría con el campo de comprensión del lenguaje natural (NLU por sus siglas en inglés). Un ejemplo clásico de esto es la diferencia al buscar en un motor de búsqueda "libros de cervantes" y "libros sobre libros de cervantes". Las palabras en si son prácticamente las mismas. Añadimos únicamente "sobre", pero eso añade un significado completamente distinto a la oración.

La última subcategoría que vamos a visitar es la de **modelado del usuario a partir de textos**, o extracción de atributos del usuario. La diferencia de estos sistemas con los anteriores es que no buscan información en el texto en sí, si no que utilizan las características del texto para inferir algún atributo del autor. Ya hemos visto trabajos en esta línea por ejemplo en (Zhang, Gao, He, & Zhou, 2016) y (Seroussi, Zukerman, & Bohnert, Collaborative Inference of Sentiments from Texts., 2010) donde utilizan la información de comentarios de texto en valoraciones para extraer atributos como la credibilidad del usuario y así modificar los ratings.

Otro ejemplo atributo del autor podría ser el tipo de personalidad de acuerdo con distintos test de personalidad, (e.g. test de Myers-Briggs o el test de "Big Five"). Un buen ejemplo de este tipo de Datasets es el dataset de Pandora (Matej Gjurković, 2020) que contiene más de 10 000 usuarios con sus respectivos comentarios, extraídos de varios subreddits. Todos los usuarios tienen asociados valores para los rasgos de personalidad, así como edad, género y nacionalidad. De esta forma se pueden entrenar modelos con algoritmos de machine learning para que sean capaces de estimar, basados únicamente en texto, los rasgos de personalidad del autor. Se podría argumentar que una desventaja de estos tipos de sistema es la necesidad de un dataset robusto con este tipo de información específica con el fin de entrenar un modelo de ML. Posiblemente este es el desafío más grande de este tipo de aplicaciones.

Llegados a este punto hemos repasado ya los distintos enfoques disponibles actualmente para la extracción de preferencias de ORSs, los puntos mejorables del trabajo inicial y, en lo relativo al ámbito de nuestra motivación, las opciones disponibles para tareas de clasificación en ML y varios tipos de caracterización y trabajo con textos en el campo de NLP. Cualquiera de estos campos tiene mucha más información y aplicaciones que no cubriremos en este trabajo ya sea por no entrar en el ámbito de este, por su complejidad o, sencillamente por su desconocimiento. No obstante, consideramos que esta es una muestra razonable de las herramientas disponibles a la hora de intentar probar nuestra hipótesis.

En las próximas dos secciones nos centraremos en la presentación de una propuesta para poner a prueba nuestra hipótesis inicial donde haremos uso de alguno (o varis) de los modelos, aplicaciones e ideas explorados en esta sección. Así mismo, se hará una presentación de la herramienta desarrollada con estos principios, así como los datos de los que partimos y el dataset que generamos.

4. Trabajo previo.

Es esta sección nos centramos en repasar el planteamiento y conclusiones del trabajo original de que partiremos (R. Centeno et al., 2018). En él se propone un nuevo mecanismo de estimación de preferencias de usuario existente en un ORS. Como ya adelantamos en la sección 2, esta aproximación evita por completo el paso de mapear preferencias en un rating numérico. En su lugar, este mecanismo estima las preferencias de un usuario dado haciendo una comparación por pares de las reviews del usuario como ya se ha visto en otros trabajos (Centeno, Hermoso, & Fasli, 2015). Pero, a diferencia de trabajos anteriores, no pide de forma activa al usuario que compare pares de objetos, ni intenta mapear el resultado de un análisis de sentimiento en un rating numérico (Hermoso, Centeno, & Fasli, 2014). En lugar de eso, en el trabajo del que partimos, los autores proponen simplemente dejar de lado esos mapeos, y utilizar directamente el valor obtenido de MPC (o análisis de sentimiento) para estimar las preferencias de usuario.

El motivo por el cual se propone esta alternativa es para ofrecer la posibilidad de ORSs que no necesiten de un rating numérico en una review para extraer las preferencias de un usuario, ya que se podrían generar *rankings de reputación individual* (IRRs) a partir de texto de la valoración exclusivamente. Esto evita, como ya hemos comentado, que los usuarios tengan que mapear sus opiniones a una escala numérica finita, perdiendo de esta forma capacidad de representación de sus preferencias. Lo cual se traduce, teóricamente, en una mejora de la precisión a la hora de determinar las preferencias de un usuario. Esto es debido a que, si la diferencia en la preferencia de dos valoraciones de un usuario es muy pequeña, esta puede convertirse en un empate a la hora de mapearla sobre una escala numérica. Para ilustrar mejor este efecto, proponemos el siguiente ejemplo:

Supongamos que tenemos al usuario u_1 y los objetos o_1, o_2 y o_3. Supongamos a su vez que el usuario a valorado estos tres objetos con las valoraciones $V_{1,1}$, $V_{1,2}$ y $V_{1,3}$ respectivamente. Por último, $e_{n,m}$ representa el valor extraído en el análisis de sentimiento de una valoración $V_{n,m}$.

Dados los siguientes valores de sentimiento en un rango [0, 10], donde a mayor valor de *e*, mayor preferencia por parte del usuario.

- $e_{1,1}$= 7.20
- $e_{1,2}$= 5.68
- $e_{1,3}$= 7.19

el modelo propuesto por (*R*. Centeno et al., 2018) ordenaría la preferencia del usuario tal que:

o1 > o3 > o2

no obstante, si en lugar de esto se opta por un modelo como el de [CITA A UNO DE ESOS DOS] donde ese valor *e* se mapea a un rating numérico, por ejemplo, en el rango [0,10] redondeando al número entero más cercano, obtenemos los siguientes $r_{n,m}$:

- $r_{1,1}$= 7
- $r_{1,2}$= 6
- $r_{1,3}$= 7

En este caso, si ordenamos por preferencia de usuario los objetos, obtenemos:

o1 = o3 > o2

Como se puede observar, el mapeo de la información extraída en el análisis de sentimiento ha distorsionado la preferencia del usuario, ahora o_1 y o_3 tienen la misma preferencia a pesar de no ser ese el caso, y la preferencia de o_2 ahora está mucho más cerca de las anteriores a pesar de tener un sentimiento de más de 1,5 por debajo de los otros dos objetos.

En el trabajo original, los autores (R. Centeno et al., 2018) realizan una comparativa de la extracción de preferencias entre métodos que realizan MPC (o análisis de sentimiento) mapeando a un rating numérico frente al método propuesto por ellos en el que se usan los valores de MPC directamente para estimar la preferencia.

Los autores del trabajo original realizan estas comparativas utilizando múltiples dataset de ORSs y distintos métodos de análisis del sentimiento para corroborar sus resultados.

En concreto, los métodos propuestos se dividen en dos familias. Por un lado, tenemos métodos basados en el uso de lexicones (en los cuales se dispone de un diccionario de palabras del lenguaje con algún valor asociado a las mismas), y, por otro lado, métodos basados en

aprendizaje automático (en los que se entrena a un sistema con textos con un valor de sentimiento ya asignado para que posteriormente sea capaz de analizar el sentimiento de textos nuevos).

En el trabajo original (R. Centeno et al., 2018) realizan una comparativa de rendimiento en mayor detalle con el mejor candidato de cada familia: Extracción de sentimiento con el método SO-CAL (*Semantic Orientation CALculation*. SOCAL de aquí en adelante) para la familia de análisis de sentimiento con lexicón; y SVR (*Support Vector Regression*, un tipo de SVM) para la familia de machine learning.

Para el experimento que proponen, utilizan 62.000 valoraciones de la base de datos de IMDb62 (de 62 usuarios distintos, 1.000 comentarios por usuario) con los que ponen a prueba la capacidad de extracción de preferencias de usuario con cada método de análisis del sentimiento. Esta base de datos representa valoraciones de usuarios a distintas películas en las que hay un valor numérico asignado por el usuario, así como un comentario de texto donde matiza su valoración. Para cada review se calcula su valor de sentimiento SOCAL y SVR. Estos dos valores se utilizan para la creación de IRR que se comparan con el IRR resultante de ordenar las reviews con la valoración numérica del usuario (como hemos comentado, incluida en la base de datos IMDb62). Cuanto más se asemejen los IRRs calculados a través de métodos de análisis de sentimiento al ranking calculado a partir de los ratings numéricos del usuario, más precisos se consideran estos IRRs.

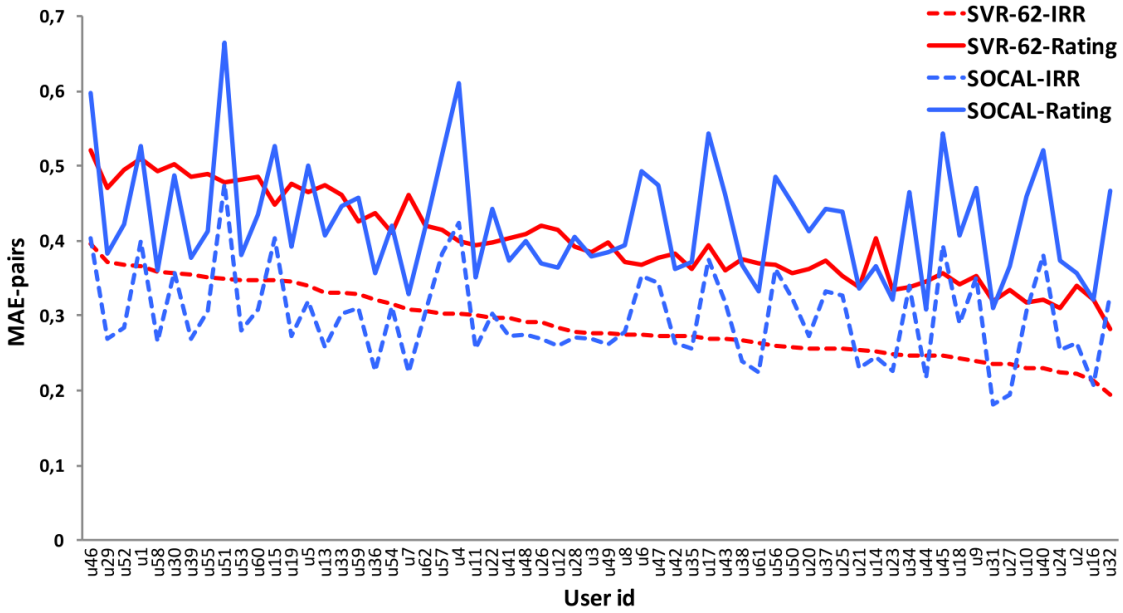


Ilustración 3: Calculo del MAE-pairs para cada usuario en el dataset IMDb62 según el método de extracción de preferencia, según (R. Centeno et al., 2018)

En la conclusión de (R. Centeno et al., 2018) se señala que, si bien ambos métodos de extracción de preferencias se acercan bastante al IRR generado por los valores del usuario, no existe un método de análisis del sentimiento que sea superior al otro para generar los IRR, sino que, según muestran los datos, cada usuario en el dataset tiene afinidad a uno de los dos métodos (i.e. algunos usuarios obtienen mejores IRR con SOCAL y otros con SVR). Esto lo podemos ver reflejado en la ilustración 3 (tomada de (R. Centeno et al., 2018)) donde se presenta, para cada uno de los 62 usuarios, el error de los IRR generados con cuatro métodos distintos de cara al IRR de valores de usuario. En nuestro caso solamente interesan las series punteadas "SOCAL-IRR" y "SVR-62-IRR". Como podemos ver, cada usuario tiene preferencia por

un método u otro (es decir, no hay un método que produzca siempre el mejor error para todos los usuarios).

5. Descripción de la propuesta.

Teniendo en cuenta nuestras hipótesis, objetivos y el punto de partida del que venimos, en esta sección se proponen los distintos mecanismos, algoritmos procesos y modelos que utilizaremos para intentar probar esas hipótesis.

Si seguimos los objetivos marcados para probar las hipótesis, el primer objetivo lo que requiere es de la implementación de un modelo de clasificador capaz de recibir una serie de datos que representen a un usuario concreto (i.e. valores extraídos de su conjunto de reviews), y una serie de categorías de métodos de análisis del sentimiento a los que se clasificará cada usuario (en nuestro caso, SOCAL y SVR). Para cumplir con este objetivo, la herramienta propuesta se ha implementado con interfaces para dar soporte a un amplio catálogo de clasificadores basados en machine learning (en teoría, adaptando un modelo de clasificador de ML cualquiera al interfaz, debería ser compatible con el sistema propuesto). Se pueden ver más detalles de la implementación especifica de este aspecto de la implementación en el apéndice A.

Este módulo de la propuesta funcionará siempre y cuando se cumpla la condición ya implícita en el objetivo 1 y que nos lleva directos al objetivo 2. Como respuesta directa a este objetivo, la herramienta propuesta implementa dos mecanismos de transformación de base de datos, en los que partimos de la base de datos original del ORS (en este caso IMDb62) y de los resultados con información de precisión de cada método de extracción del sentimiento por usuario. Los procedimientos descritos en la sección A.1 del apéndice A describen en detalle el proceso, pero para tener una idea general, se describirá brevemente aquí:

La generación del dataset definitivo se divide en tres partes, aunque en la práctica dos de ellas se realizan juntas siempre. La primera se corresponde a la creación de un dataset que combine la información de los usuarios junto con el método de análisis de sentimiento que mejor les funcione (i.e. SOCAL o SVR). Al final de este paso disponemos de usuarios con la clase a la que esperamos ser capaces de clasificarlos y todos los comentarios de texto de sus reviews, más los ratings numéricos originales provistos por el usuario.

El segundo paso implica la creación de lo que denominaremos “instancias de usuario”.

Una instancia de usuario representa una concatenación de una cantidad *n* de textos de reviews de usuario para generar lo que sería caso de review de este usuario “nueva”. De tal modo, que, según la parametrización del sistema, este creará *n* instancias de cada usuario con una longitud (i.e. cantidad de comentarios de texto concatenados) de valor *m*.

Finalmente, el último paso implica recorrer una a una todas las instancias de usuario generadas en nuestra base de datos y generar para cada una de ellas todos los atributos con información del texto/autores implementados en la herramienta. Con estos dos módulos implementados se satisfacen los dos primeros objetivos de nuestra propuesta. Pero para poder si quiera probarlos, es necesario tener implementados los generadores de atributos que comentaremos en las próximas secciones.

Adicionalmente, en el apéndice A se puede encontrar una descripción de la tecnología con la que se implementará la herramienta, así como la especificación de la estructura de la base

de datos de la herramienta en la que se irán guardando los resultados y datos intermedios generados.

5.1. Atributos propuestos para la clasificación de instancias de usuario.

En esta sección se presenta el núcleo de nuestra propuesta. En otras palabras, las formas de las que extraemos información de los textos de la base de datos en forma de atributos, para después evaluar su utilidad a la hora de clasificar los usuarios. La implementación de estos métodos de generación de atributos satisface directamente los objetivos 3 y 4 vistos en la sección 2, y sin ellos no hay trabajo. De ahí que este sea el núcleo de nuestra propuesta.

Se han implementado tres métodos de extracción de atributos tras considerar las opciones disponibles y la naturaleza del problema. Como primera aproximación, se ha elegido un tipo de Sentence Embedding, en concreto, el modelo propuesto en (Nils Reimers, 2019) usando la librería que facilitan, Sentence_transformers.

A partir de ahí, se proponen dos alternativas. La idea en estas propuestas buscar atributos que tengan relación con el modo de funcionamiento de los dos métodos de extracción de sentimiento utilizados en (R. Centeno et al., 2018). En el caso de SOCAL, este está basado en la asignación de valores a las palabras de un texto sin tener en cuenta ni estructura sintáctica ni semántica. Con esto en mente, se proponen una serie de atributos de carácter estadístico-morfológico que resuman el tipo de categorías gramaticales utilizadas en el texto.

Finalmente, dada la naturaleza de caja negra de los modelos de machine learning como SVR, no se ha podido plantear un tipo de atributo o atributos relacionados con su funcionamiento. Esta falta de información respecto a su funcionamiento interno nos permite probar una aproximación distinta sin preocuparnos demasiado con la afinidad a SVR. Hasta ahora se ha presentado una propuesta que tiene en cuenta la semántica (Sentence-Bert) y otra que tiene en cuenta la morfología. Como última propuesta se ha escogido el enfoque de modelado de usuarios a partir de textos haciendo uso del dataset de Pandora (Matej Gjurković, 2020). A continuación, describimos en más detalle las tres propuestas.

5.1.1. Vector de Sentence Embedding con Sentence-BERT.

Esta opción de atributo genera un vector de 768 valores reales a partir del conjunto de textos de la instancia de utilizando el módulo Sentence_transformers de la librería de Sentence-BERT (Nils Reimers, 2019).

Estos vectores codifican la similitud semántica, o lo parecidos que son dos textos en base a su posición en el espacio de 768 dimensiones. Esta opción se ha implementado bajo la hipótesis de que los usuarios de una clase tienen vectores en una zona del espacio N-dimensional bien definida y fácil de delimitar. La principal razón del uso de este tipo de atributo (entiéndase Sentence embeddings) es el éxito que demuestran en tareas de comprimir grandes cantidades de información semántica de un texto en el vector, lo que nos lleva a pensar que son el principal candidato para encontrar características que separen las clases, aunque nosotros no podamos entender cuáles son.

5.1.2. Atributos estadísticos morfológicos con el POS-tagger de NLTK.

Estos atributos, como puede indicar el nombre, tienen en cuenta frecuencias de aparición de distintas categorías morfológicas. Esta opción de generación de atributos es la más sencilla de las tres disponibles.

Esta opción de generación de atributos es la más simple de las tres propuestas, ya que no implica el uso de mecanismos complicados para calcular los atributos, simplemente hace un conteo de etiquetas. Para ello, primero se extraen valores estadísticos de la morfología o la sintaxis del texto.

Los atributos que extraemos, a partir del conjunto de reviews de una instancia de usuario son:

- Media de palabras por review.
- Media de sustantivos por review.
- Media de adjetivos por review.
- Media de oraciones por review.

Estos se generan mediante el etiquetado de las categorías gramaticales del texto (*Part-Of-Speech-tagging*) utilizando la librería NTLK (NLTK Project, s.f.). Tras etiquetar el texto de la instancia se cuentan las apariciones de las categorías gramaticales que buscamos y se dividen entre en número de reviews de la instancia en cuestión.

5.1.3. Atributos de modelado del usuario a partir de textos con el dataset de Pandora.

En esta opción se han utilizado técnicas para el modelado del usuario a partir de texto con el fin de extraer una serie de atributos que aludan no al texto en sí, si no al autor. Los atributos que se generan son los siguientes:

- Valores de personalidad del test Myers-Briggs: Este test propone la representación de la personalidad de un sujeto en base a 4 baremos. Los extremos de cada baremo son siguientes:
    - Extroversión / Introversión.
    - Sensorial / Intuitivo.
    - Racional / Emocional.
    - Calificador / Perceptivo.
- Estimación de la edad del usuario.
- Estimación del género del usuario.

El test Myers-Briggs (MBT de ahora en adelante) se representa como un vector de cuatro números en el rango del 0 al 1 donde el 0 es un extremo de la escala y el 1 es el otro (e.g. si el primer valor es 0, el usuario es extrovertido, pero si es 1, el usuario es introvertido). Estos cuatro valores se pueden obtener como valores discretos (el usuario es extrovertido o introvertido, pero no hay término medio), o como un valor continuo real.

De forma similar, la estimación de genero también se puede obtener de forma discreta (0 masculino, 1 femenino) o como un número real entre 0 y 1 que señalaría la probabilidad de que el autor tuviera un género u otro (>0.5 femenino, <0.5 masculino, por ejemplo).

Finalmente, el valor de la edad siempre se devuelve como un número real positivo.

Para obtener estos atributos, se ha utilizado el dataset de Pandora (Matej Gjurković, 2020) que contiene 17,640,062 comentarios de 10,288 sacados de múltiples subreddits. En este dataset, 9,084 usuarios tienen asociados sus valores de MBT, así como su edad y su género.

Gracias a esto, se ha conseguido entrenar a tres clasificadores con este dataset para que cada uno calcule un tipo de atributo relacionado con el autor. Los atributos se dividen entre los

clasificadores como se ha visto más arriba. Un clasificador para el MBT, otro para la edad, y un último para el género. De este grupo de atributos se extraen cinco configuraciones diferentes combinando las opciones que tenemos:

- MBT con valores discretos (vector de cuatro enteros).
- MBT con valores continuos (vector de cuatro reales).
- Edad y Genero con valor discreto (vector de real y entero).
- Edad y Genero con valor continuo (vector de dos reales).
- MBT, Edad y Genero con valores continuos. (vector de seis reales).

En los experimentos, a la hora de utilizar atributos de modelado de usuario, elegiremos una de estas cinco opciones.

6. Diseño experimental.

En esta sección se tomará la herramienta y atributos propuestos en el apartado anterior y se usarán para poner a prueba nuestra hipótesis. Para ello, se proponen distintas configuraciones para la herramienta, así como las métricas necesarias para evaluar el rendimiento de dichas configuraciones.

En primer lugar, se define una configuración como la serie valores elegidos para los parámetros de la herramienta. Como ya se ha mencionado en la sección anterior, el sistema tiene principalmente cuatro parámetros con los que vamos a experimentar. Estos son los siguientes:

- El vector de atributos utilizado como entrada para la clasificación de un usuario.
- El modelo de clasificador utilizado para asignar clase a los usuarios.
- La distribución de instancias de cada clase en el dataset.
- La cantidad de textos que componen las instancias de usuario en el set de validación.

Debido a la motivación principal del trabajo (i.e. averiguar si es posible clasificar a los usuarios según sus comentarios y ver qué información se puede extraer de los mismos), de los cuatro parámetros presentados, los que más se alinean con nuestro objetivo son la elección de vector de atributos y la elección de clasificador. El primero nos dirá, en teoría, que información es relevante en los comentarios de texto. El segundo parámetro determinará si es posible clasificar a los usuarios en base a la información que tenemos y los modelos de clasificación disponibles.

No obstante, esto no les resta importancia a los parámetros restantes, ya que la distribución de clases en el dataset nos indicará la tolerancia de los modelos a datasets muy desequilibrados a la hora de entrenarlos. Por otro lado, la longitud de las instancias de usuario nos dirá como de aplicable es este sistema en un escenario realista, ya que un sistema entrenado que necesite veinte mil valoraciones para poder clasificar a un usuario nuevo no es un sistema realista.

En lo que se refiera a que clasificadores elegir, nuestro criterio tiene en cuenta sobre todo el rendimiento sobre la transparencia. Esto se debe a que en primer lugar queremos saber si es posible clasificar. La manera en la que se clasifica es una pregunta para futuros trabajos. Por lo tanto, hemos elegido una serie de modelos de clasificador populares disponibles en Scikit-learn, centrándonos en que obtengan buenos resultados de clasificación y sean rápidos.

Para cada configuración que se pruebe, se necesita una métrica (o métricas) que ayude a evaluar si dada configuración mejora la creación de IRR a partir de texto o no. En las próximas secciones se presentan estas métricas, entre ellas el cálculo de MAE-pairs, los porcentajes de acierto, y la simulación de creación de IRR.

6.1. Datasets.

Para la creación del clasificador, partimos de el dataset de reviews de películas de IMDB, compuesto de 62 usuarios y 1000 reviews por usuario.

Reviews	Usuarios	Casos SOCAL	Casos SVR
62 000	62	13	49

Tabla 1: Composición del dataset a clasificar.

Este set es uno de los utilizados por (R. Centeno et al., 2018) a la hora de generar los IRR. Se cuenta también con los resultados de su trabajo, en otras palabras, un set donde se dispone de los valores de sentimiento para cada review tanto con SOCAL como con SVR, además de la puntuación asignada manualmente por el usuario en la review.

Antes de empezar con los experimentos necesitamos procesar estos dos datasets (el original con los comentarios, y el resultante del trabajo de (R. Centeno et al., 2018) con los IRRs). Nuestro objetivo es generar una base de datos única que relacione tanto los valores de los IRRs como los comentarios textuales con la tabla maestra de usuarios. Esa base de datos será nuestro punto de partida.

Para ello, tenemos que utilizar la herramienta en su primer modo de funcionamiento: lectura de Datasets de partida. De esta forma, se unen los dos dataset (el original con los textos de reviews y el generado por (R. Centeno et al., 2018) con los valores de sentimiento) en un único dataset centralizado en la base de datos.

Posteriormente, para facilitar el tratamiento de los resultados más adelante, se debe calcular la clase a la que pertenece cada usuario de la base de datos. Para ello se utiliza la misma métrica que proponen en (R. Centeno et al., 2018): MAE-pairs (*Mean Averaged Error-pairs*). Esta métrica comprueba la preferencia de todas las parejas de reviews posibles en el conjunto de reviews de un usuario, usando como indicador de la preferencia el valor de SOCAL o SVR (según que MAE-pairs se quiera calcular).

Las preferencias de estas parejas se comparan con las preferencias resultantes de utilizar los valores numéricos asignados por el usuario (valores disponibles en el dataset original). De esta forma, cuantas más parejas coincidan en preferencia, más se acerca el IRR generado con un método de extracción del sentimiento al IRR del usuario. MAE-pairs indica entonces la cantidad de parejas mal ordenadas entre la cantidad total de posibles parejas en el set de reviews del usuario. A continuación, podemos ver el pseudocódigo para el cálculo del MAE-pairs.

Una vez calculado el MAE-pairs de SOCAL y de SVR para cada usuario, la clase del usuario corresponderá al método que tenga menor MAE-pairs.

```
Parejas_erroneas = 0
Parejas_totales = 0
Para cada uc en U:
        Para cada uc2 no comparado con uc
                Parejas_totales++
                Si (uc.rating > uc2.rating) es distinto a (uc.socal > uc2.socal)
                        parejas_erroneas++
                fin si
        fin bucle
Fin bucle
Return parejas_erroneas/parejas_totales
```

Pseudocódigo 1: Calculo del MAE-pairs-socal para un usuario U.

El siguiente paso es el de la creación de las instancias de usuario que vamos a clasificar. Una instancia de usuario representa una concatenación de valoraciones escogidas de forma aleatoria del conjunto de valoraciones de ese usuario. El objetivo de esto es aumentar el tamaño de nuestra base de datos de cara al entrenamiento de los modelos y tener instancias de usuario con distintos tamaños que nos permita evaluar el impacto de la cantidad de valoraciones de un usuario en su clasificación.

De este modo, se crearán para cada usuario 4000 instancias de usuario (i.e. un usuario estará representado 4000 veces en la base de datos, cada una de ellas con una cadena de texto distinta). Es importante que estos textos de instancia se elijan al azar para que los vectores de atributos de cada instancia puedan ser distintos. Nuestro dataset de instancias estará compuesto de 248000 instancias de usuario al terminar este paso.

Nuestro último paso para obtener el dataset del cual nos serviremos para experimentar es el de generación de los vectores de atributos. Este es seguramente el punto más costoso en cuanto a recursos computacionales se refiere. Esto se debe a que se deben generar todos los vectores de atributos para cada una de las instancias de usuario, y algunas de ellas como los Sentence embeddings consumen muchos recursos para ello. Una vez hayamos terminado la generación y almacenado los vectores en la base de datos, nuestro dataset estará listo para su uso.

6.2. Parámetros del sistema.

En esta sección se profundiza en las distintas opciones de que disponemos para cada uno de los parámetros del sistema mencionados previamente. Se hace hincapié en los candidatos a atributo de una instancia de usuario.

6.2.1. Clasificadores.

Como se ha mencionado en secciones previas, nuestro criterio a la hora de elegir un clasificador del repertorio que tenemos es que clasifique bien y clasifique rápido. Ya que nuestra motivación es saber si es posible o no clasificar, otras consideraciones como la transparencia del clasificador no son de importancia en este momento.

Los modelos de clasificador elegidos como candidatos (de la librería Scikit-learn) en base a esos criterios son los siguientes:

- LogisticRegression.
- MLPClassifier (Perceptrón Multi-Capa).
- SVC (de la familia de *Support Vector Machine).*
- Ridge.
- DecisionTreeClassifier.
- ExtraTreesClassifier.
- RandomForestClassifier.

Como se puede ver, hay representantes de múltiples familias de modelos de clasificación. La gran mayoría pertenece a la rama de modelos basados en error.

6.2.2. Distribución de clases en el dataset de instancias de usuario.

Este parámetro hace referencia a la distribución de instancias de cada clase en el dataset de instancias de usuario. De los datos de la Tabla 1 se observa que, aproximadamente el 79,03% de los usuarios pertenecen a la clase SVR y el 20.97% restante a la clase SOCAL. Esta proporción de casos no es ideal para la tarea de aprendizaje, ya que se corre el riesgo de que el clasificador converja en la clase mayoritaria (i.e. la clase asignada a una instancia de usuario sea siempre SVR).

De este modo, la configuración de este parámetro implica dejar fuera del ciclo de entrenamiento y validación cierto porcentaje de instancias de la clase mayoritaria (en este problema, SVC). En otras palabras, este parámetro nos permite manipular la relación de clases en el dataset de instancias de usuario. Dicho esto, actualmente solo se ha implementado la posibilidad de convertirlo en un dataset equilibrado al 50/50.

Para nuestros experimentos se van a considerar el dataset completo, con el porcentaje actual de casos, y una versión en la que se excluyan los suficientes casos SVR para alcanzar un porcentaje del 50% para cada clase.

6.2.3. Atributos de representación de las instancias de usuario.

En esta sección se presentan las distintas alternativas de vectores de atributos para los experimentos. A la hora de implementar estos vectores en el sistema, partimos de los distintos atributos propuestos en la sección 4 (i.e. Sentence embeddings, estadístico-morfológicos, y atributos de usuario). Sin embargo, dentro de los atributos de usuario que nos ofrece el dataset de (Matej Gjurković, 2020) podemos hacer combinaciones que nos resulten interesantes. De este modo, aplicamos el dataset pandora para generar un vector con atributos de usuario con valores que evalúen cada categoría del test de Myers-Briggs, otro con estimaciones de edad y género, y un último con la combinación de estos dos. Cabe señalar que estos vectores generados con pandora pueden devolvernos valores discretos (clasificando cada categoría como un 0 o un 1) o valores continuos en ese rango. Con todo esto en mente, los vectores de atributos implementados finalmente son los que siguen:

- Sentence Embedding (BERT).
- Métricas estadístico-morfológicas (EM).
- Myers-Briggs Test Discreto (MBTd).
- Myers-Briggs Test Continuo (MBTc).
- Estimación de Edad y Genero Discreto (EGd).
- Estimación de Edad y Genero Continuo (EGc).
- Pandora completo (MBT Continuo + Edad y Genero Continuo).

6.2.4. Cantidad de review en las instancias del set de validación.

Este parámetro del sistema tiene como objetivo determinar la cantidad mínima de texto que necesitamos de un usuario para que los atributos generados en la sección anterior sirvan para predecir la clase del usuario. Las opciones para este parámetro son, al dividir el dataset de instancias de usuario, previo al entrenamiento del clasificador, crear un set de validación que contenga únicamente instancias de la longitud X dada, usando el resto de las instancias del dataset para el entrenamiento. Las distintas longitudes de instancia para las que hemos generado atributos son:

- Instancias de 10 reviews
- Instancias de 100 reviews
- Instancias de 500 reviews
- Instancias de 1000 reviews (todas las reviews de un usuario sin excepción)

Para este parámetro nos interesan sobre todo las instancias con menor número de reviews (10 y 100). El efecto de este parámetro se evaluará como en el caso de los atributos a través de las métricas que veremos en la próxima sección.

6.3. Métricas de los resultados.

En esta sección se presentan las distintas métricas que se utilizarán para la evaluación de los resultados del sistema con cualquier configuración utilizada (se entiende por configuración cualquier combinación exclusiva de parámetros del sistema). Estas métricas corresponden a dos familias. La primera es la medición de la precisión a la hora de clasificar un dataset de validación con el clasificador elegido. Hablaremos más de esta familia en la siguiente sección. La segunda se trata de una simulación del sistema de (R. Centeno et al., 2018), con los datos de que disponemos, en la que se compara, por cada usuario, el MAE-p resultante de generar el IRR con SOCAL (siempre), el MAE-p de SVR, y el MAE-p de cada usuario según nuestro clasificador indique que dicho usuario debe usar SVR o SOCAL. El valor ideal para esta métrica es que, para cada usuario, el clasificador siempre elija el método con menor MAE-p.

6.3.1. Medición de porcentaje de precisión en validación.

Este tipo de medición es muy sencillo. Consiste en clasificar una serie de casos de los cuales conocemos la clase esperada, comparar con la salida del clasificador, y contar los aciertos.

El número de aciertos dividido entre el número de casos nos da la precisión. No obstante, esta medida depende del set que se utiliza para la validación. Si los casos de validación se hubieran utilizado en el entrenamiento, la red estaría clasificando casos que ya “conoce”, por lo tanto, el resultado de la validación queda comprometido, así como el porcentaje de precisión.

Por eso mismo, se han implementado dos algoritmos para calcular la precisión de un experimento.

El primero de ellos es el famoso k-fold-cross-validation. Este algoritmo divide el dataset en *k* subsets (o folds), y repite el ciclo de entrenamiento->validación k veces. En cada ciclo elije un fold distinto como set de validación, y utiliza los k-1 folds restantes para el entrenamiento. De este modo, al terminar los k ciclos el algoritmo ha validado el sistema con todos los casos del dataset, pero en el momento de validación, ningún caso era conocido por el clasificador de antemano. El porcentaje final es la suma de aciertos en cada ciclo dividido entre el número (Thorsten Hennig-Thurau, 2004) (Cui, 2012)ro de casos en el dataset.

El segundo algoritmo implementado se ha hecho a medida del parámetro de la sección (6.2.4. Cantidad de review en las instancias del set de validación). Este divide el dataset en un set de validación exclusivamente con instancias de una longitud N determinada, y utiliza el resto de las instancias para el entrenamiento del clasificador. El porcentaje de acierto en este caso no refleja al dataset completo como en k-fold-cross-validation, si no que refleja únicamente a las instancias de longitud N.

6.3.2. Simulación de generación de IRR.

Esta métrica nos muestra de forma muy visual la mejora que supondría utilizar nuestro sistema para la predicción del método de generación de IRR frente a lo propuesto en (R. Centeno et al., 2018), que supondría utilizar SOCAL para todos los usuarios o SVR para todos los usuarios. En esta métrica, un resultado bueno significaría que, para cada usuario, se elige siempre el método de generación de IRR con menor MAE-pairs. La grafica resultante es similar a la Ilustración 3 añadiendo una serie más que correspondería al resultado de utilizar nuestro método.

Una nota de cara a esta métrica es que, para cada usuario, el método (o clase) que se elige, es el resultado de ponderar la clasificación de todas las instancias de ese usuario. En otras palabras, si un usuario tiene 4 instancias en el dataset de validación, tres de ellas socal (representadas con el valor 1) y una de ellas SVR (representada con el valor 0), la clase media elegida para ese usuario en la configuración de la red utilizada 0,75. Si este valor es mayor que 0,5, se considera que ese usuario se clasifica por norma general como SOCAL, de lo contrario, se le asigna SVR.

7. Resultados experimentales.

En esta sección se definen la serie de experimentos a realizar teniendo en cuenta los parámetros a nuestra disposición para configurar el sistema y las métricas de las que disponemos para evaluar los resultados. La idea principal de estos experimentos es determinar, en primer lugar, si es posible clasificar a los usuarios con cualquiera de los vectores de atributos implementados. Esto contestaría directamente a nuestra hipótesis principal, que en cascada contaría al resto de hipótesis. Adicionalmente, se busca jugar con los parámetros para encontrar una configuración que maximice la precisión del sistema.

Por lo general esto se va a llevar a cabo evaluando el rendimiento de los siete vectores de atributos ya presentados usando distintas configuraciones. A continuación, se presentan estos experimentos.

7.1. Resultados de Comparación de Clasificadores.

Este primer experimento queremos lanzar la prueba más simple. Se trata de intentar descartar los clasificadores que no parezcan funcionar bien con una serie de pruebas superficiales. Por un lado, este experimento puede contestarnos a la cuestión de la viabilidad de clasificar usuarios. Por otro lado, se trata de reducir la cantidad de tests "innecesarios" con clasificadores que, a primera vista, no parecen rendir tan bien.

Para ello, se quiere comparar la precisión los distintos clasificadores enumerados en (6.2.1. Clasificadores) utilizando el valor devuelto por 10-fold-cross-validation para su comparación, y utilizando el dataset de instancias de usuario tal y como está, es decir, con proporciones de clases desbalanceadas (i.e. 79% SVR, 21% SOCAL).

Esta prueba se quiere realizar con cada uno de los vectores de atributos implementados. Esto nos da siete configuraciones por clasificador (i.e. EM, BERT, MBTid, MBTic, ESd, ESc, PandoraCompleto). En otras palabras, un total de 49 experimentos individuales que representarían todas las combinaciones clasificador-vector de atributos usando únicamente el dataset desbalanceado.

A continuación, podemos ver los resultados en la siguiente tabla:

	Sentence BERT	Estadístico-morfológico	MBTi Discreto	MBTi Continuo
DecisionTreeClassifier	70,16%	97,19%	79,03%	79,20%
ExtraTreesClassifier	81,09%	97,78%	79,03%	85,19%
LogisticRegression	77,37%	80,38%	79,03%	79,03%
MLPClassifier	78,35%	80,35%	79,03%	79,03%
RandomForestClassifier	81,35%	97,53%	79,03%	84,73%
Ridge	81,26%	80,65%	79,03%	79,03%
SVC	80,13%	79,03%	79,03%	79,03%

	Edad y Genero Continuo	Edad y Genero Discreto	Pandora Completo
DecisionTreeClassifier	76,96%	71,48%	86,05%
ExtraTreesClassifier	81,09%	71,92%	91,59%
LogisticRegression	79,03%	79,03%	79,03%
MLPClassifier	79,03%	79,03%	79,03%
RandomForestClassifier	81,33%	71,85%	90,54%
Ridge	79,03%	79,03%	79,03%
SVC	79,03%	79,03%	79,03%

Tabla 2: Resumen de las tasas de acierto para los siete vectores de atributos según el clasificador utilizado

Lo primero que hay que tener en cuenta al interpretar estos resultados es la distribución del dataset de instancias. Debido a que la clase mayoritaria (i.e. SVR) representa

aproximadamente el 79% del dataset, cualquier clasificador que este cerca de ese porcentaje de acierto estará, probablemente, clasificando todas las instancias del dataset como SVR. Por lo tanto, se consideraría como mejora todo clasificador que tenga una tasa de acierto mayor que el porcentaje de la clase mayoritaria. Eso significaría que, sin lugar a duda, es capaz de clasificar y clasificar correctamente.

Dicho esto, se puede observar que, por lo general, los mejores clasificadores para cada vector son RandomForestClassifier y ExtraTreesClassifier. La excepción seria Edad y genero discreto, para el cual estos dos clasificadores empeoran el comportamiento, y MBTi Discreto, que no consigue buenos resultados con ningún clasificador.

Una última observación que se puede hacer sobre los resultados de la tabla es que, dentro de los vectores de atributos de usuario (las 5 últimas columnas), los vectores que utilizan valores discretos (0 o 1) funcionan considerablemente peor que los valores continuos.

Por otro lado, ya desde este experimento podemos ver que, partiendo de los datos originales a nuestra disposición (i.e. dataset de IMDB y resultados de (R. Centeno et al., 2018)), es posible asignar de forma satisfactoria métodos de análisis de sentimiento a los usuarios basándonos en sus comentarios Elaboraremos más al respecto en la sección 7.

7.2. Resultados de Comparación de proporción del dataset.

Este experimento tiene como objetivo evaluar que impacto tiene en la tarea de aprendizaje la distribución de clases en el dataset. Así mismo se desea ver si la elección de vector de atributos tiene algún efecto o si todos reaccionan igual al cambio en la distribución. Tras evaluar los resultados del experimento anterior, hemos decidido basarnos en el modelo de clasificador RandomForestClassifier para llevar a cabo este experimento y los siguientes. Esto se debe a que consideramos que la información que puedan aportar en este punto el resto de los clasificadores no nos ayuda con nuestro objetivo original.

Para ello, se realizará una validación (i.e. se ejecutarán el modelo elegido usando 10-fold-cross-validation) de cada uno de los siete vectores de atributos dos veces. La primera utilizando el dataset con la proporción que tiene por defecto (i.e. 79/21 SVR y SOCAL respectivamente), y la segunda podando las suficientes instancias de la clase mayoritaria para que el dataset resultante este equilibrado al 50/50.

Se mostrará el resultado para todos los vectores de atributos en una tabla que reflejará tanto la precisión obtenida, como la mejora en clasificación frente al modelo original de creación de IRRs (i.e. asignar estáticamente a todo el dataset el método mayoritario, en este caso SVR). Esto, teóricamente representa la mejora en la elección de método de análisis de sentimiento en los IRRs frente al trabajo original.

Se han añadido dos columnas con el prefijo “Mejora”. Estas columnas muestran como una configuración de clasificador, dataset y vector de atributos concreta mejoran el valor del caso base, que sería elegir de forma inmutable la clase mayoritaria para todas las instancias de usuario. En el caso del dataset con distribución desequilibrada, este caso base sería elegir siempre SVR, mientras que en el dataset equilibrado el caso sería elegir siempre una clase, sea la que sea, ya que no hay clase mayoritaria.

De esta forma estamos comparando los IRRs generados por nuestro sistema con los IRRs generados por el trabajo original de (R. Centeno et al., 2018) eligiendo siempre SVR como método de análisis del sentimiento.

La precisión de clasificación se mide realizando 10-fold-cross-validation con el dataset, tal y como hemos hecho en el experimento anterior. Podemos ver los resultados a continuación.

Vector de Atributos	Proporcion 79/21	Proporcion 50/50	Mejora 79/21	Mejora 50/50
Sentence BERT	80,96%	63,64%	1,93%	13,64%
Estadístico-morfológico	97,53%	98,14%	18,50%	48,14%
MBTi Discreto	79,03%	58,10%	0,00%	8,10%
MBTi Continuo	84,73%	78,90%	5,70%	28,90%
Edad y Genero Continuo	81,33%	81,00%	2,30%	31,00%
Edad y Genero Discreto	71,85%	69,99%	-7,18%	19,99%
Pandora Completo	90,54%	94,58%	11,51%	44,58%

Tabla 3: Resumen de las tasas de acierto para los siete vectores de atributos según la distribución de clases en el dataset utilizando el clasificador RandomForestClassifier

A primera vista, parece que usar la proporción 50/50 implica una pérdida de precisión en la clasificación para prácticamente todas las configuraciones de vectores. No obstante, hay que tener en cuenta la mejora de clasificación de cara a los casos base (i.e. que solo se clasifique SVR). Una vez tenemos eso en cuenta, vemos que la mejora en clasificación es mayor para un dataset que está distribuido al 50/50. En otras palabras, un dataset equilibrado nos da un modelo de clasificación con mayor mejora en clasificación que su contraparte entrenada con un dataset desequilibrado. Esto en teoría se traduciría en un mejor funcionamiento a la larga con usuarios nuevos ya que hay menos riesgo de que el clasificador se haga sobreentrenado en la clase mayoritaria.

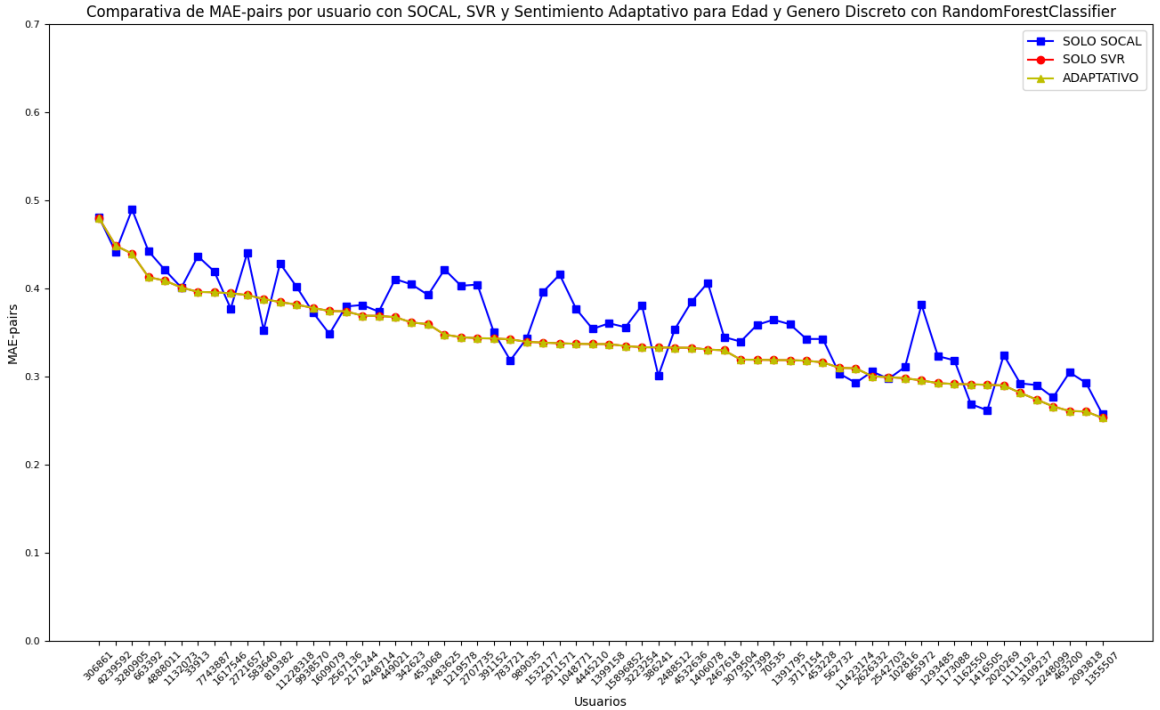


Ilustración 4: Simulación del cálculo de IRR para los 62 usuarios con los tres métodos: Siempre SVR, siempre SOCAL, y método adaptativo con clasificador RandomForestClassifier, vector de atributos de edad y genero discretos (EGd) y dataset no equilibrado.

Adicionalmente, se puede observar que Edad y Genero Discreto, no se va a un valor de clasificación por defecto (clasificar todas las instancias como SVR) cuando usamos el dataset con la distribución por defecto, sino que, de hecho, clasifica incorrectamente aún más casos.

Ya que se considera un caso curioso (la red hace un esfuerzo en clasificar mal), mostramos a continuación la simulación de IRR para esa configuración.

En este grafico () podemos ver que clase le correspondería a cada usuario (eje x), de media (i.e. teniendo en cuenta el resultado de clasificación de todas las instancias de ese usuario) usando RandomForestClassifier y la proporción de instancias inicial. Podemos ver que nuestro método con esta configuración de atributos se ajusta siempre a la clase mayoritaria, pero en tal caso el porcentaje de acierto debería ser 79%, no un 72%. Esto se debe a que la clase elegida en la ilustración para cada usuario es la media de las clases asignadas a cada instancia de ese usuario (recordemos que cada usuario está representado por 4000 instancias en el dataset). Existen más fallos de clasificación que los observables en la gráfica, pero al ponderar los resultados de clasificación de las instancias, se pierde información.

Veamos un ejemplo práctico. Si un usuario tiene cuatro instancias de clasificación $\{i_1; i_2; i_3; i_4\}$ a las que se le asignan las clases {1; 1; 1; 0} respectivamente (0 para SOCAL, 1 para SVR en este ejemplo), la clase promedio para dicho usuario será 0.75. Debido a que la separación de las dos clases está en 0,5, y al ser la clase promedio mayor que este valor, se le asignara clase SVR, quedando así ocultos los casos de clasificación incorrecta. Sin embargo, esa clasificación incorrecta se tiene en cuenta de cara a la precisión global de esa configuración del modelo de clasificador, de ahí que obtengamos resultados peores al caso de elegir siempre la clase mayoritaria, pero esto no se muestre en la .

Con el objetivo visualizar mejor la información oculta de esta gráfica, a continuación, se muestra la clase promedio para cada usuario en la siguiente tabla:

Usuario	Clase Media	Usuario	Clase Media
306861	0,1515	1399158	0,1515
8239592	0,3333	15896852	0,1667
3280905	0,2879	3223254	0,2424
663392	0,1212	386241	0,4697
4888011	0,1667	2488512	0,0909
1132073	0,3182	4532636	0,1212
33913	0,1061	1406078	0,1667
7743887	0,1667	2467618	0,2273
1617546	0,2727	3079504	0,3030
2721657	0,1061	317399	0,1970
583640	0,3030	70535	0,1667
819382	0,1667	1391795	0,1515
11228318	0,0455	3717154	0,1364
9938570	0,4091	453228	0,4545
1609079	0,2879	562732	0,2879
2567136	0,1818	11423174	0,4848
2171244	0,3333	2626332	0,1818
4248714	0,1818	2542703	0,2879
449021	0,0758	102816	0,1970
342623	0,1212	865972	0,0152
453068	0,2576	1293485	0,0909
2483625	0,1212	1173088	0,1970
1219578	0,2576	1162550	0,3485
2707735	0,1515	1416505	0,4242
391152	0,2121	2020269	0,1061
783721	0,2879	1111192	0,2424
989035	0,1061	3109237	0,2576
1532177	0,2273	2248099	0,3485
2911571	0,1364	463200	0,1364
1048771	0,1364	2093818	0,2879
4445210	0,1970	1355507	0,1364

Tabla 4: Clasificación media de los 62 usuarios de IMDb62 utilizando RamdomForestClassifier con el vector Edad y Genero Discreto (EGd) y sin equilibrar las clases del dataset de instancias de usuario.

Lo que se puede ver en esta tabla es que todos los usuarios tienen cierta cantidad de instancias clasificadas erróneamente, pocos tienen una media de clase SVR con menos de un 0.10 de media y varios usuarios se han clasificado en la gráfica como SVR por puro azar, ya que vemos que la media de clasificación está muy cerca del 0.5 que separa las dos clases.

Para poner esta información en un contexto mayor, veamos que aspecto tiene tanto la gráfica de clasificación como la clase promedio de cada usuario cuando repetimos el experimento, pero utilizamos el vector de atributos estadístico-morfológico:

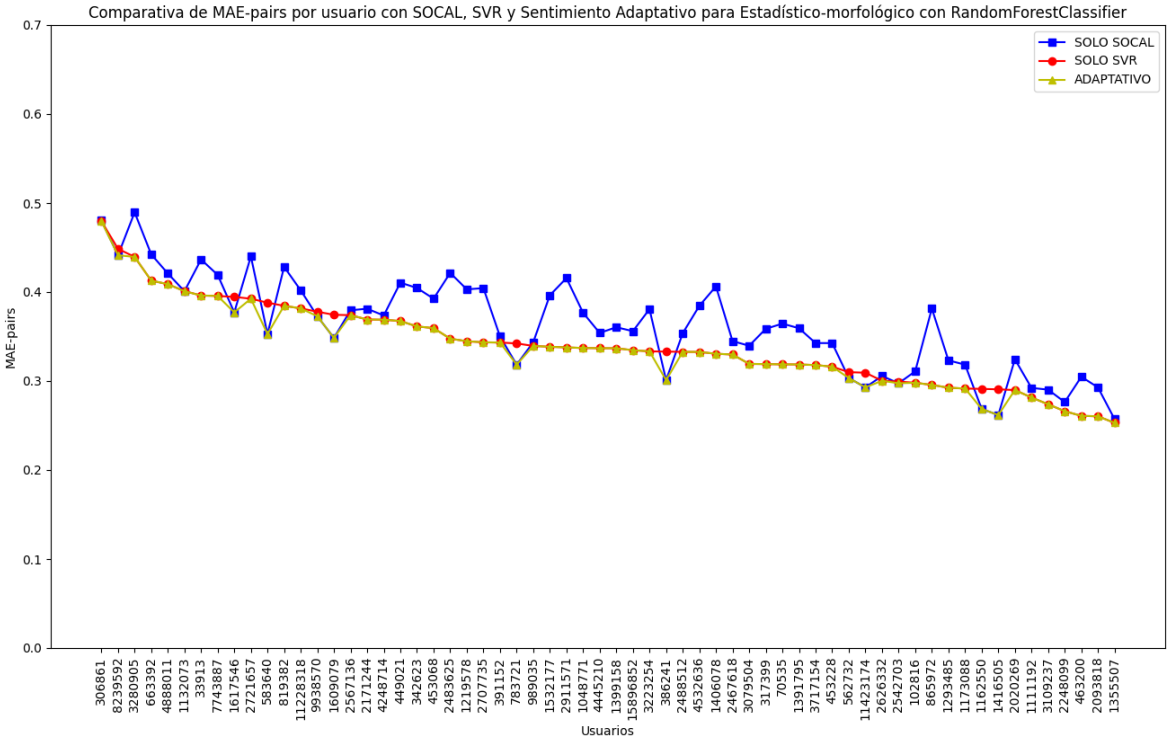


Ilustración 5: Simulación del cálculo de IRR para los 62 usuarios con los tres métodos: Siempre SVR, siempre SOCAL, y método adaptativo con RandomForestClassifier, vector estadístico-morfológico (EM) de atributos y dataset no equilibrado.

Como se puede observar en la ilustración 5, a diferencia de la configuración anterior, la configuración con vector estadístico-morfológico cambia el método de clasificación para cada usuario de forma satisfactoria. De hecho, elige bien en los 62 casos.

Inicialmente puede parecer que esto choca con la información en la Tabla 4, no obstante, hay que tener en cuenta que la clase elegida que estamos viendo para cada usuario es una media de la clase asignada a todas sus instancias, de modo que, tal y como ocurría en el caso anterior, hay fallos en clasificación que quedan ocultos por el promedio. Esto se puede ver mejor en la tabla que, como la Tabla 6, contiene la clase promedio para cada uno de los usuarios calculada a partir de la clasificación de todas sus instancias.

Usuario	Clase Media	Usuario	Clase Media
306861	0,0152	1399158	0,0000
8239592	1,0000	15896852	0,0000
3280905	0,0152	3223254	0,0000
663392	0,0606	386241	0,9697
4888011	0,0455	2488512	0,0303
1132073	0,9394	4532636	0,0000
33913	0,0606	1406078	0,0152
7743887	0,0000	2467618	0,0152
1617546	0,9091	3079504	0,0000
2721657	0,0152	317399	0,0000
583640	0,9545	70535	0,0000
819382	0,0606	1391795	0,0000
11228318	0,0000	3717154	0,0000
9938570	0,9242	453228	0,0152
1609079	0,9242	562732	0,8030
2567136	0,0000	11423174	0,9545
2171244	0,0455	2626332	0,0000
4248714	0,0000	2542703	0,8788
449021	0,0000	102816	0,0152
342623	0,0000	865972	0,0152
453068	0,0152	1293485	0,0000
2483625	0,0000	1173088	0,0152
1219578	0,0000	1162550	0,9848
2707735	0,0152	1416505	0,8939
391152	0,0000	2020269	0,0000
783721	1,0000	1111192	0,0000
989035	0,0758	3109237	0,0000
1532177	0,0000	2248099	0,0000
2911571	0,0000	463200	0,0000
1048771	0,0000	2093818	0,0000
4445210	0,0152	1355507	0,0152

Tabla 5: Clasificación media de los 62 usuarios de IMDb62 utilizando RamdomForestClassifier con el vector de atributos estadístico-morfológico con una distribución desequilibrada de clases de instancias de usuario

Si comparamos esta tabla con la Tabla 4 en la que tenemos las clases promedio para el vector de atributos EGd, lo primero que salta a la vista es que las clases promedio del vector de atributos estadístico-morfológico están mucho más polarizadas. Podemos ver muy claramente si un usuario pertenece a la clase 0 (SVR en estos resultados) o a la clase 1 (SOCAL). De hecho, existen muchos casos de usuarios en los que todas sus instancias se han clasificado correctamente, ya que hay clases promedio con valor 1 y valor 0.

De este análisis podemos concluir que el vector de atributos de Edad y Genero discreto no genera una separación limpia del conjunto de usuarios en dos clases. En su lugar genera una separación que se sale del caso que entendemos sería el más sencillo (i.e. asignar siempre la clase mayoritaria) y consigue clasificar incorrectamente más casos. La razón de esto se desconoce.

7.3. Resultados de Comparación de sets de validación con distinta longitud de instancia.

Este tercer experimento está diseñado para poner a prueba los límites del sistema de clasificación de los usuarios.

Queremos evaluar el efecto en la longitud (en número de reviews concatenadas) de las instancias que componen el set de validación de cara a la clasificación de los vectores de atributos. En otras palabras, lo que intentamos averiguar en este experimento es saber, **una vez el modelo se ha entrenado completamente**, cuantos comentarios/reviews textuales necesitamos por parte de un usuario para que la clasificación de este sea fiable. Así mismo, queremos ver si los distintos vectores de atributos dependen de igual manera de la cantidad de reviews disponibles para clasificar al usuario.

Para poner un ejemplo más sencillo, imaginemos un modelo completamente entrenado y que empezamos a utilizar de forma práctica. La pregunta que surge es ¿Cuántas valoraciones debe realizar un usuario antes de que se le asigne un método de análisis del sentimiento optimo? Es posible que necesite solo una valoración (lo cual sería el caso ideal), o puede que necesite más de 100 (lo cual no es del todo realista para el usuario promedio). De ahí surge nuestro interés de intentar acotar la cantidad mínima de valoraciones de usuario necesarias.

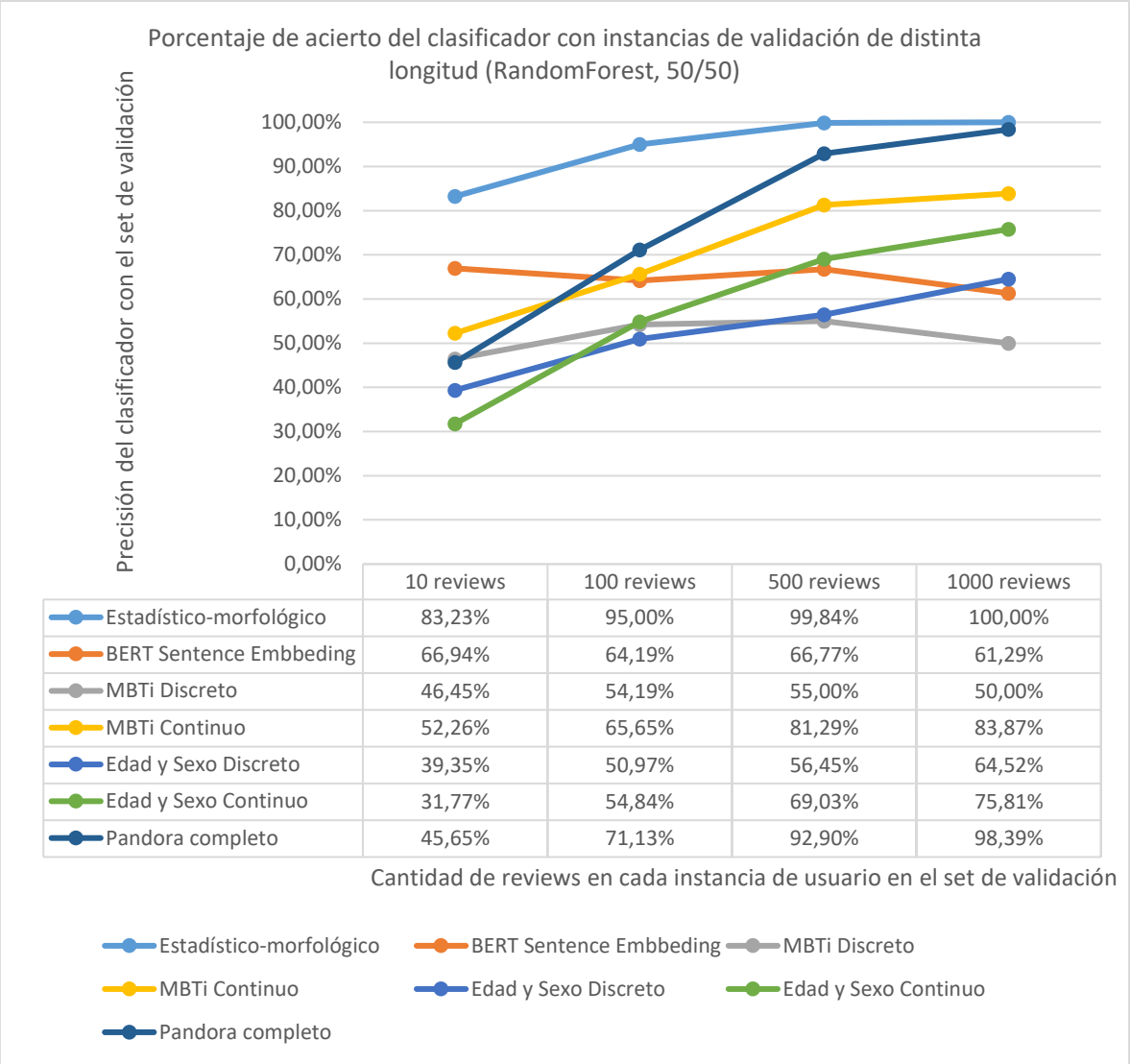
Para ello, se realizará una validación de cada uno de los siete vectores de atributos con cuatro sets de validación compuestos por instancias de usuario con distintas cantidades de reviews concatenadas. Los números de reviews por instancia para los cuatro sets de validación son los siguientes:

- Set 1: Instancias de usuario de 10 reviews.
- Set 2: Instancias de usuario de 100 reviews.
- Set 3: Instancias de usuario de 500 reviews.
- Set 4: Instancias de usuario de 1000 reviews.

Se presentará un gráfico con los resultados obtenidos con cada uno de los siete vectores de atributos y utilizaremos de nuevo el modelo de clasificador elegido en base a los resultados del primer experimento. Del mismo modo, también nos centraremos exclusivamente en mostrar los resultados usando la distribución de dataset que mejores resultados obtenga en el segundo experimento. Esto se debe a que nuestra motivación es averiguar si es posible clasificar a los usuarios, y, de manera secundaria, que información tienen más relevancia dentro de sus valoraciones. Debido a estas dos motivaciones, nuestra prioridad es ver el comportamiento de todos los vectores de atributos en las mejores configuraciones del sistema que vamos encontrando a lo largo de los experimentos.

Como parámetros del experimento, hemos entrenado el modelo de RandomForestClassifier con una distribución de clases equilibrada al 50/50 y lo hemos validado utilizando instancias de usuario que contengan 10 reviews, 100, 500 y 1000 (todas las reviews de ese usuario). Nuestra expectativa aquí es ver como a mayor cantidad de reviews, el usuario es clasificado con más precisión.

A continuación, podemos ver un resumen de los resultados de este experimento para los siete vectores de atributos implementados:



	10 reviews	100 reviews	500 reviews	1000 reviews
Estadístico-morfológico	83,23%	95,00%	99,84%	100,00%
BERT Sentence Embbeding	66,94%	64,19%	66,77%	61,29%
MBTi Discreto	46,45%	54,19%	55,00%	50,00%
MBTi Continuo	52,26%	65,65%	81,29%	83,87%
Edad y Sexo Discreto	39,35%	50,97%	56,45%	64,52%
Edad y Sexo Continuo	31,77%	54,84%	69,03%	75,81%
Pandora completo	45,65%	71,13%	92,90%	98,39%

Ilustración 6: Porcentaje de acierto en la clasificación de usuarios con instancias de usuario de longitud distinta (en cantidad de reviews). Modelo de clasificación RandomForestClassifier con distribución de clases 50/50.

Antes de empezar a evaluar el grafico, hay que recordar que cualquier resultado en el 50% de precisión o cerca del mismo probablemente signifique que el clasificador está clasificando todo como SOCAL o como SVR. Esto se debe a que estamos usando un dataset equilibrado.

Con esto en mente, vemos que la mayoría de los vectores de atributos cumplen con nuestra expectativa de mejorar el rendimiento según aumenta la cantidad de reviews, con dos excepciones claras: BERT y Myers-Briggs discreto. En el caso de BERT podemos ver que empieza clasificando entorno a un 67% de precisión, pero este valor va bajando conforme aumentamos la cantidad de reviews. Esto puede deberse a cómo funciona el modelo de Sentence embeddings. Recordemos que un Embedding de este tipo sitúa una cadena de texto en un espacio n-dimensional, donde su relación o similitud a otra cadena de texto viene dada por su proximidad en ese espacio a esa otra cadena.

Debido a que estos vectores a menudo son compuestos a partir de los vectores de los Word embeddings individuales de cada palabra del texto, nuestra hipótesis es que es posible que cuando más extenso sea el texto, su posición en el espacio n-dimensional vaya convergiendo

al mismo punto para todos los usuarios, haciéndolos cada vez más difíciles de distinguir. No obstante, habría que profundizar mucho en este caso para poder corroborar o desmentir esta hipótesis. En este punto es simplemente especulación por nuestra parte.

Por otro lado, tenemos el caso de MBTi discreto. En este caso vemos que el porcentaje de acierto en la clasificación sencillamente está bailando alrededor del 50%. Lo único que nos indica esto es que este vector de atributos es incapaz de dividir el conjunto de usuarios en las dos clases que tenemos. Este dato es consistente con el resto de los experimentos realizados donde vemos que MBTi discreto está siempre cerca de la clase mayoritaria o del 50% dependiendo de que distribución de dataset utilicemos.

Si nos fijamos en el resto de los vectores, queda bastante claro que los dos mejores vectores de atributos para la tarea de clasificar los usuarios según el método de análisis de sentimiento son: Vector Pandora Completo (compuesto de pandora MBT Continuo y Edad y Genero Continuo) y vector estadístico-morfológico.

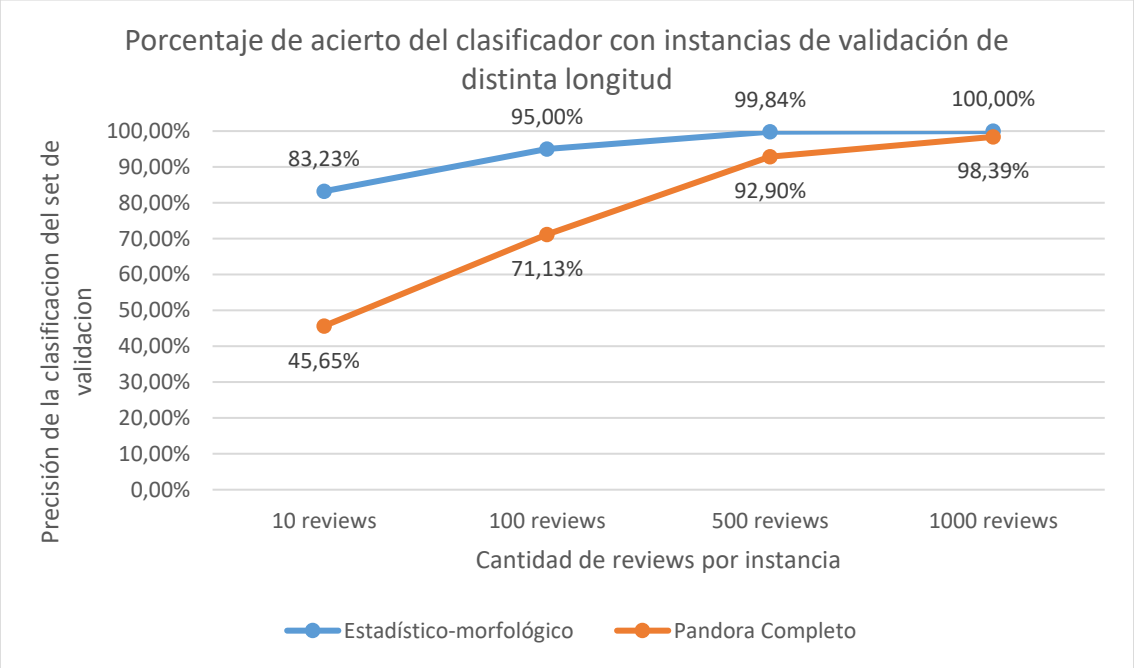


Ilustración 7: Comparativa de la precisión del clasificador utilizando atributos estadístico-morfológicos y todos los atributos (continuos) obtenidos gracias al dataset PANDORA, para sets de validación con distinta longitud de instancia

En este grafico hemos sacado a parte sus resultados (también presentes en el grafico anterior) para repasarlos con menos interferencia. Estos dos ejemplos cumplen nuestra expectativa inicial de mejora en la clasificación a mayor número de reviews. Adicionalmente, se puede observar que el vector estadístico-morfológico consigue resultados de clasificación superiores al 90% con tan solo 100 reviews, mientras que Pandora Completo necesita al menos 500 reviews por instancia para llegar a ese nivel de precisión aproximado. No es la primera vez a lo largo de los experimentos realizados en los que el resultado del vector más sencillo (en nuestra opinión) es el mejor de todos. Esto es algo que queremos explorar en profundidad e intentar explicar ya que no es un resultado esperado originalmente.

7.4. Comentarios sobre la aproximación con atributos estadístico-morfológicos.

Como comentábamos en la conclusión del experimento anterior, algo que nos ha llamado la atención a lo largo de los experimentos es como los atributos morfológicos, a pesar

de su simplicidad, hayan conseguido unos niveles de precisión tan altos. Por lo tanto, hemos considerado procedente realizar un pequeño análisis de los datos generado para ver si existe una fuerte división entre los atributos para casos de SOCAL y casos de SVR.

En primer lugar, se ha realizado un sencillo análisis estadístico para cada uno de los atributos en cada uno de los textos de usuario según la clase a la que pertenezcan (SOCAL y SVR). En la siguiente tabla vemos los valores medios y la desviación típica para cada atributo en cada clase.

Atributo	Valor medio	Desviación típica	Clase
ADJ_COUNT	30,077	5,470	socal
ADJ_COUNT	31,016	5,566	svr
NOUN_COUNT	70,959	8,369	socal
NOUN_COUNT	80,168	8,935	svr
SENT_COUNT	12,740	3,501	socal
SENT_COUNT	14,514	3,806	svr
WORD_COUNT	287,566	16,954	socal
WORD_COUNT	324,534	18,013	svr

Tabla 6: Cantidad de ocurrencias promedio de un atributo concreto y desviación típica por clase (SVR o SOCAL)

Aquí ya hay un indicio que apunta a cierta separación de las clases, se puede ver que el valor medio para todos los atributos es **siempre menor** para la clase SOCAL que para SVR. Con esto en mente, lo siguiente que se ha hecho es comprobar la distribución de frecuencia para cada atributo en SOCAL y SVR.

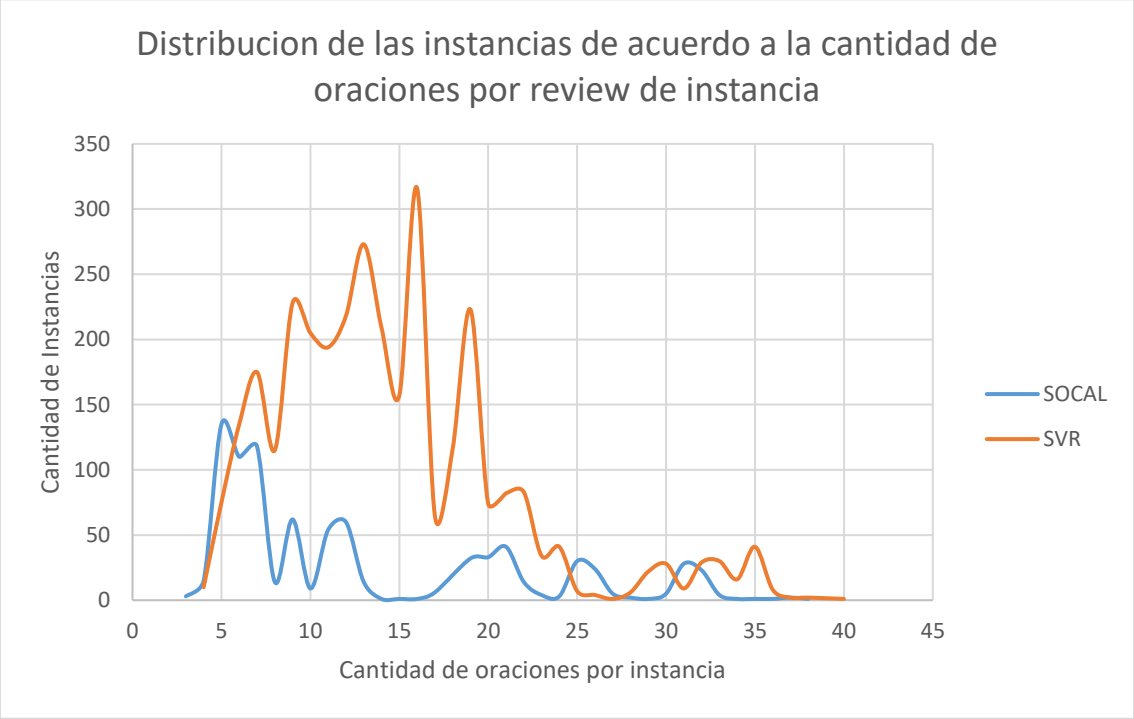


Ilustración 8: Distribución de las instancias de usuario de acuerdo con la cantidad de oraciones por review de instancia

Este grafico muestra la cantidad de instancias de usuario de una clase determinada que tiene X oraciones. En el podemos ver que las instancias de SOCAL se concentran a la izquierda con un menor número de oraciones de media. Esto se repite en los otros tres tipos de atributo como veremos a continuación.

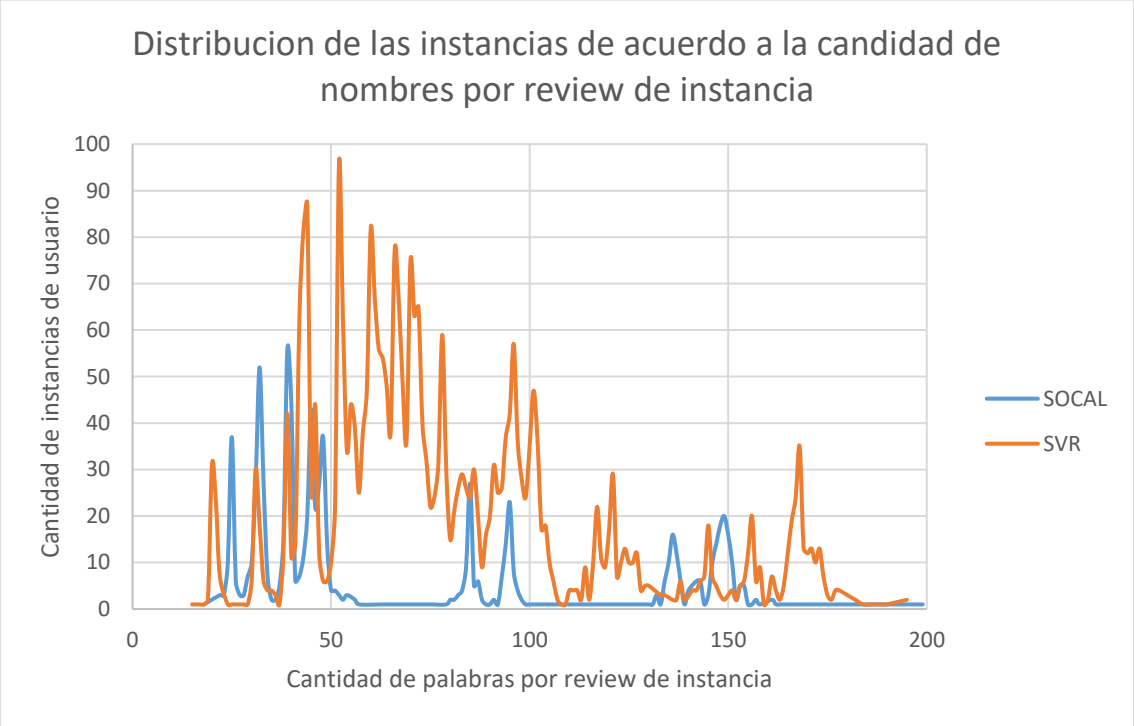


Ilustración 9: Distribución de las instancias de acuerdo con la cantidad de nombres por review de instancia

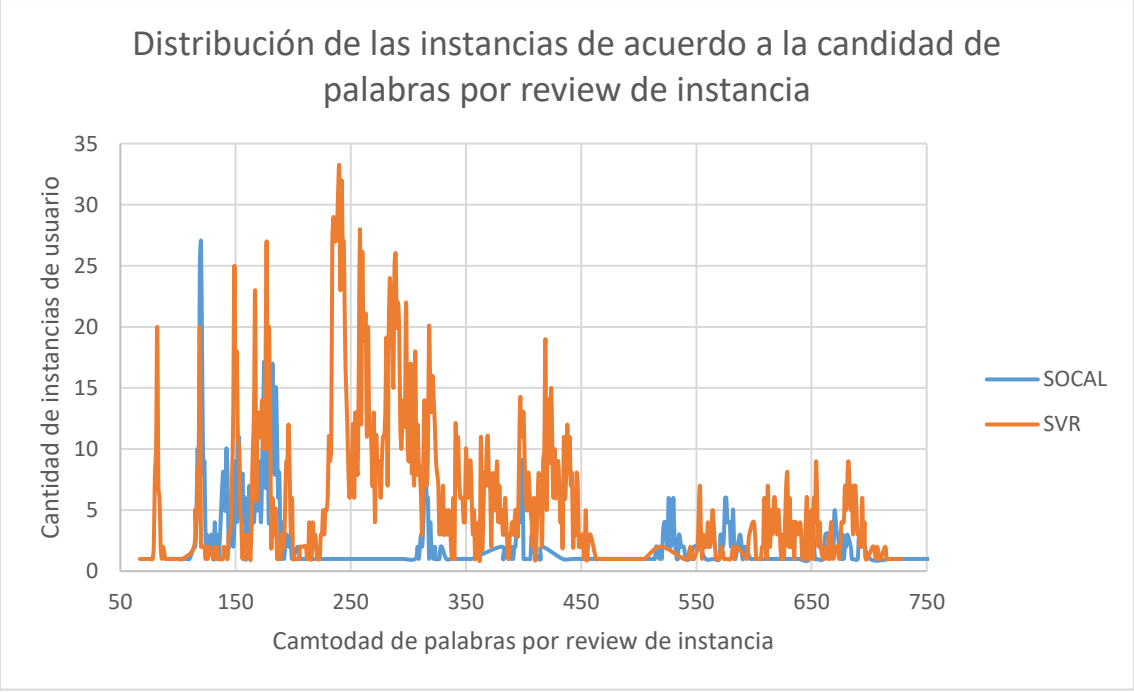


Ilustración 10: Distribución de instancias por cantidad de palabras por review de instancia

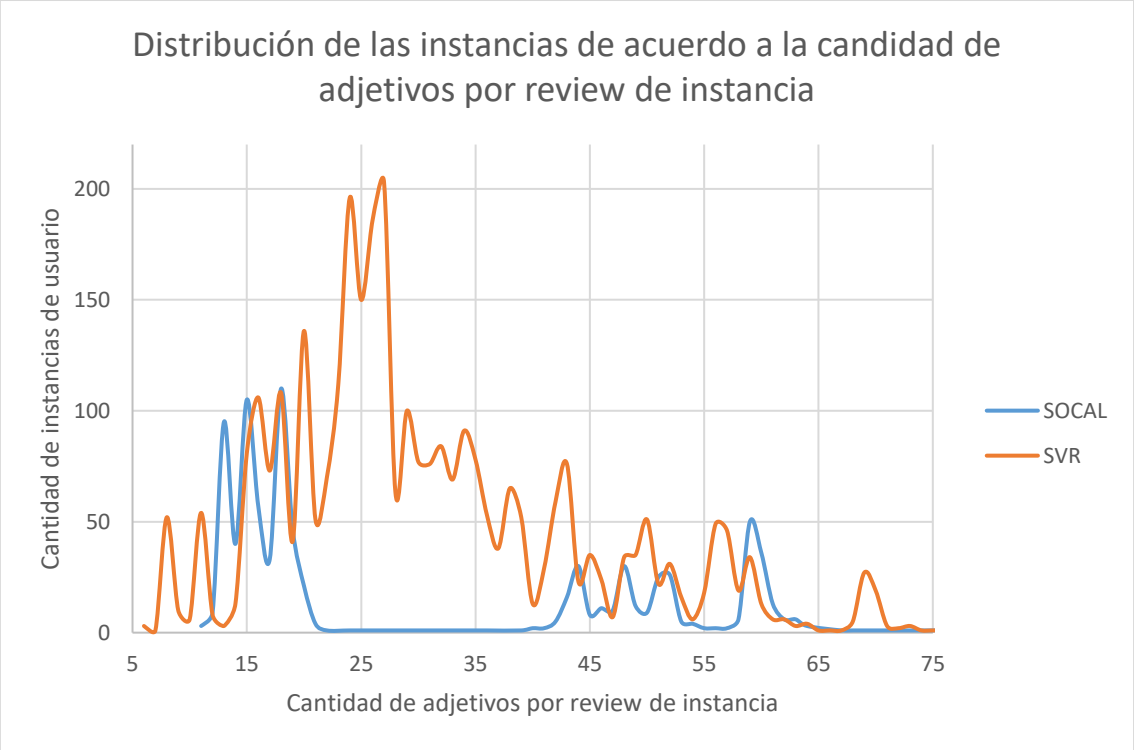


Ilustración 11: Distribución de las instancias de acuerdo con la cantidad de adjetivos por review de instancia

Lo que parecen indicar estos datos es que, a mayor longitud media de las reviews de un usuario, la eficacia de SOCAL disminuye rápidamente. Esto puede deberse a que el enfoque basado en lexicón de SOCAL no tenga una buena tolerancia para términos no contenidos en su catálogo. Sería necesario profundizar en los mecanismos internos de SOCAL para determinar con más seguridad a que puede deberse esto, pero la cantidad de texto está definitivamente relacionada.

8. Conclusiones.

En ese trabajo hemos propuesto mejoras al modelo de extracción de preferencias de usuario por medio de la creación de IRRs presentado en el trabajo de (R. Centeno et al., 2018). En primer lugar, hemos analizado los resultados que obtuvieron y hemos señalado la posibilidad de escoger métodos análisis del sentimiento para los usuarios de forma dinámica en el momento de la creación de sus IRRs. Esto mejoraría en última instancia la precisión con la que se extraen las preferencias de usuario del ORS. Nuestro objetivo principal, derivado de esto, ha sido el de implementar un sistema capaz de clasificar los distintos usuarios del ORS asignándoles uno de los dos métodos de análisis del sentimiento disponibles, SOCAL o SVR. Para ver si esta tarea era posible, se ha implementado una herramienta capaz de generar, para el dataset de usuarios del ORS original, una serie de vectores de atributos a partir de las reviews de los propios usuarios. Estos vectores contienen información o bien relativa a los comentarios de texto a partir de los cuales se han creado, o bien relativas al autor de esos comentarios de texto.

Con la herramienta implementada y el dataset generado, los resultados de la primera prueba en la sección anterior ya muestran que el sistema es capaz de clasificar los usuarios entre nuestras dos clases posibles (SOCAL y SVR) con distintos modelos de clasificación. Debido a que en este trabajo nos interesaba más saber si esto era posible, no se ha profundizado en las

razones por las que unos clasificadores han funcionado mejor que otros. Ya según los resultados de este primer experimento podríamos concluir que la hipótesis de que es posible clasificar a los usuarios según su método de MPC es correcta, al menos con algunos de los vectores de atributos propuestos en este trabajo (Especialmente, el vector de atributos estadístico-morfológicos y el vector de Pandora Completo).

Extendemos en este punto nuestros objetivos para hacer un estudio del comportamiento del sistema frente a distintos parámetros disponibles para el mismo. Estos son la distribución de clases del dataset de entrada de cara al entrenamiento del modelo de clasificación, y el comportamiento del sistema a la hora de clasificar usuarios para los que hay disponibles distintos volúmenes de reviews. Con el primero de estos parámetros no parece haber grandes sorpresas, ya que se observa que con un dataset de entrenamiento debidamente balanceado la mejora a la hora de clasificar por parte del modelo es mayor que con un dataset mal distribuido. Esto se alinea con las expectativas generales de modelos de machine learning. Por otro lado, el estudio de la capacidad de clasificación de usuarios del sistema cuando el usuario en si tiene pocas reviews con las que clasificarle nos muestra que, dependiendo de que vector de atributos es utilizado, el sistema puede llegar a necesitar cientos de reviews de un usuario para clasificarle con cierto grado de fiabilidad. En este aspecto, el vector de atributos que mejor funcionan (con diferencia) es de nuevo el vector estadístico-morfológico.

Debido los buenos resultados de este vector, hemos ido un paso más allá he hemos hecho un breve análisis estadístico de la estructura de estos vectores según a qué clase de método MPC pertenezca ese usuario. A falta de un estudio en mayor profundidad, nuestra conclusión es que, por alguna razón, la característica de los textos de usuario que divide a los usuarios en las dos clases es la longitud media de sus comentarios.

En resumen, se parte en este trabajo para conseguir mejorar la precisión con la que se estiman las preferencias de usuario en un ORS mediante la asignación de forma dinámica de un método de análisis del sentimiento a cada usuario en el momento de crear su IRR. Tras implementar todos los sistemas, herramientas y datasets necesarios, llegamos a la conclusión de que esto es posible, y de que, teóricamente, nuestro sistema mejora la precisión de la extracción de las preferencias de los usuarios de la aproximación presentada en el trabajo del que partimos.

8.1. Trabajo futuro.

De este trabajo nos surgen varias ideas de vías de investigación que podrían resultar interesantes, si bien algunas pueden tener más utilidad que otras.

En primer lugar, el siguiente paso natural a este trabajo sería la integración con el trabajo original de (R. Centeno et al., 2018) para poder corroborar los resultados obtenidos sobre el dataset de IMDB con otros similares (o no tan similares). Esto por un lado nos ayudaría a ver posibles carencias o mejoras a la solución presentada aquí, y por el otro nos daría una herramienta de generación de IRRs mejorada como subproducto.

En segundo lugar, y siguiendo con la motivación de corroborar los resultados obtenidos, sería interesante implementar métodos de análisis de sentimiento con la esperanza de que sean competitivos a la hora de generar IRRs y así tengamos un escenario donde se pueda repetir nuestro análisis, pero cada usuario tenga más posibles opciones a la hora de ser clasificado.

Otra posible vía de trabajo sería profundizar en las razones por las que el vector de atributos aparentemente más sencillo (estadístico-morfológico) es el que ha obtenido mejores

resultados. Este trabajo parece señalar a que lo que en realidad afecta es la longitud del texto en sí mismo, pero sería necesaria más investigación para corroborarlo.

Así mismo, uno de los problemas del trabajo realizado es la opacidad de los métodos de clasificación utilizados. Estos se han escogido por el volumen de datos con el que se trabaja, pero sería interesante utilizar un método de clasificación más transparente para poder interpretar mejor las razones por las que un vector de atributos funciona mejor que otro. Esto, potencialmente, nos permitiría comprender mejor cual es la información relevante en los comentarios de los usuarios, mejorando la generación de los IRRs, y, en última instancia, la extracción de preferencias.

De cara a medir la precisión de esta solución, consideramos de interés también realizar pruebas con distintas cantidades de comentarios en la fase de entrenamiento. En otras palabras, querríamos saber cómo de extenso debe ser el dataset inicial para generar un modelo de clasificador que pueda llevar a cabo la tarea de asignar métodos de análisis de sentimiento a los usuarios. Así mismo, no estaría de más llevar esta idea al extremo y ver cómo se comportan los mejores vectores (como Pandora completo y estadístico-morfológico) cuando el modelo se entrena con pocas reviews, y se clasifica también con pocas reviews.

Finalmente, con motivo de encontrar más información relevante en los comentarios, se considera recomendable experimentar con más tipos de atributos extraíbles del texto con la esperanza de encontrar atributos que funcionen igual de bien que los estadístico-morfológicos, o los embeddings de Bert, pero además sean transparentes de tal forma que podamos entender por qué son relevantes.

Apéndice A: Implementación de la herramienta de software

En este apéndice ofrecemos una visión más detallada de la herramienta de software desarrollada en Python para llevar a cabo la experimentación propuesta en este trabajo. Dividimos el apéndice en varias secciones. Las secciones A.1 y A.2 definen los procesos mediante los cuales se transforman los datasets originales de los que partimos (i.e. dataset IMDB y el dataset de resultados del trabajo original) para obtener un dataset con instancias de usuario y sus respectivos vectores de atributos, listos para ser alimentados a un modelo de clasificador. En la sección A.3 hacemos referencia a librerías y toolkits de los que nos hemos ayudado con el fin de implementar nuestra herramienta. La sección A.4 ofrece una vista a alto nivel de la arquitectura interna de la herramienta, y finalmente la sección A.5 representa la estructura de la base de datos en la que almacenamos el dataset final.

A.1. Proceso de generación del dataset de atributos.

En esta sección vemos en mayor detalle el proceso que seguimos para la generación de los atributos que hemos visto en la sección anterior dado un texto.

La generación de atributos está estructurada de tal manera que facilite la implementación y consecuente integración de un nuevo generador de atributo en la herramienta. Con este fin en mente, la clase VoidAttrGen (Ilustración 13) funciona a modo de interfaz para implementar nuevos generadores. Cualquier generador debe contener las funciones get_attr(text) y get_attr_id () para que el motor de generación de atributos los gestione de forma transparente.

El proceso es el que sigue: si la herramienta se ejecuta en el modo de funcionamiento 3 (generación de atributos), el motor de generación de atributos carga una instancia de cada

generador marcado como activo en la base de datos, utilizando para ello los identificadores de atributos en la base de datos en una función de mapeo que recibe el id y devuelve el generador. Una vez se cargan todos los generadores en el motor de generación, este recibe una lista de instancias de usuario. Para cada instancia, se comprueba que atributos ya han sido generados y que atributos aún no se han calculado, haciendo uso del método get_attr_id() del generador que está siendo evaluado y buscando esa ID en la lista de atributos de la instancia de usuario. Se calculan únicamente los atributos que aún no están presentes en la instancia con get_attr(texto), y a continuación la instancia se actualiza en la base de datos. Al terminar con todas las instancias, termina el proceso de generación de atributos.

En caso de crear un nuevo generador, se debe añadir a la tabla de maestro para que el motor lo cargue a la hora de generar atributos. Se pueden omitir generadores que ya no interesan si se marca el atributo como desactivado en la base de datos.

A.2. Proceso de clasificación de instancias de usuario.

A estas alturas, disponiendo en la base de datos de una serie de instancias de usuarios (de la longitud que sean), la herramienta ya puede operar en el modo 4, clasificación de usuarios.

El proceso de clasificación es el siguiente: La herramienta carga de la base de datos las instancias con los atributos necesarios según el id_setup que se facilite. En la tabla de NNSETUPS encontrará la consulta SQL para la configuración de atributos a utilizar. Una vez cargados, se escoge uno de los modelos de machine learning contenidos en la librería Scikit-learn.

Con el modelo escogido, se divide el set de instancias cargadas de la base de datos en kfolds, y por cada fold resultante se realiza un ciclo de entrenamiento del modelo (entrenamiento con k-1folds y validación la precisión con 1 fold). En este proceso, durante la validación de cada fold, se le asigna a la instancia de usuario el valor estimado por el clasificador, de esta forma, para cada instancia en memoria tenemos tanto la clase a la que pertenece (SOCAL o SVR) como la clase que predice el clasificador.

Una vez terminado el proceso de clasificación, se graban los resultados en RESULTS_CLASSIFICATION en el formato (user_id, error-socal, error-svr, error-adaptativo, nrevs) siendo error adaptativo el error resultante al crear un IRR con el método dictado por el clasificador y nrevs la longitud (en reviews) de la instancia de usuario.

Se han realizado pruebas con siete clasificadores distintos dentro de la librería de Scikit-learn. Se presentan más adelante.

A.3. Herramientas utilizadas.

A continuación, se repasan todas las herramientas y tecnologías utilizadas (i.e. lenguajes de programación, librerías utilizadas, bases de datos, etc.).

Desde el punto de vista del lenguaje elegido, se ha utilizado principalmente Python. Esto se debe a que tiene un ecosistema con una gran variedad de librerías extremadamente útiles que engloban la mayoría de los recursos necesarios para implementar nuestra solución. Entre ellas encontramos:

- Natural Language Toolkit (NLTK Project, s.f.): Esta librería contiene un amplio repertorio de funciones que permiten trabajar con textos para tareas de tokenización, clasificación, stemming, **POS-tagging**, etc.
- Sqlite3 (AOL developers, s.f.): Librería con funciones para la gestión de una base de datos relacional de tipo SQLite
- Scikit-learn (Pedregosa, 2011): Esta librería está compuesta de distintos modelos de machine learning para propósitos de clasificación, regresión, etc.
- Matplotlib: Librería utilizada principalmente para la generación de gráficos y soporte visual para datos.
- Sentence_transformers (Nils Reimers, 2019): Esta librería ofrece de Pytorch ofrece métodos para la creación de Sentence embeddings basados en el modelo de BERT.

A.4. Arquitectura.

La herramienta propuesta se divide en tres módulos diferenciados. El primero de ellos, el módulo de entrada y salida de datos. Encargado de la lectura de ficheros, y lectura/escritura en la base de datos. En segundo lugar, está el módulo de generación de atributos. En ese modulo podemos encontrar todas las clases que componen el motor de generación de atributos para instancias de texto. Finalmente está el módulo principal, que coordina la comunicación entre los dos módulos anteriores y alberga cualquier función adicional que no encaje en ninguno de los anteriores.

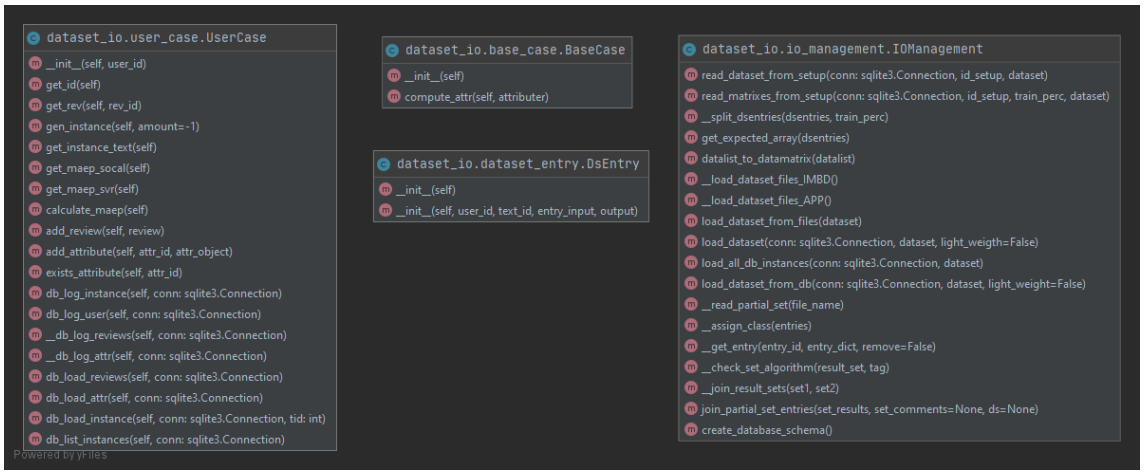


Ilustración 12: Módulo de entrada/salida con los métodos necesarios para leer los sets de fichero y para leer y escribir en la base de datos de la herramienta.

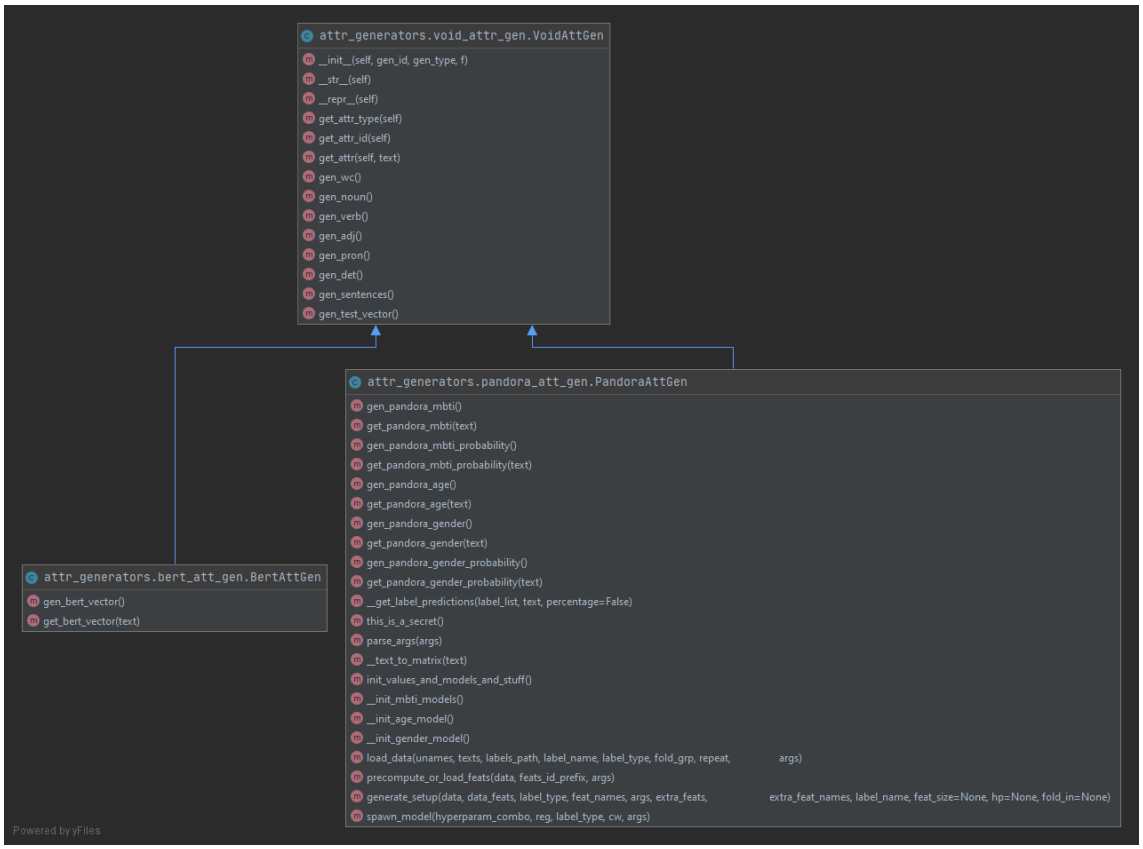


Ilustración 13: Módulo de generación de atributos, con las clases y métodos necesarios para la generación de atributos de un texto en el módulo principal

La herramienta tiene cuatro modos de funcionamiento distintos:

1. Lectura de los dataset de partida, es decir las colecciones de usuarios que queremos evaluar y clasificar, según nos las proporciona (R. Centeno et al., 2018), y almacenamiento del dataset en nuestra base de datos

2. Generación de instancias de usuario. Una instancia de usuario es un objeto que contiene el identificador de usuario y una cadena de texto que corresponde a la concatenación de N de sus reviews al azar. Esto es necesario ya que en nuestra propuesta el clasificador recibe usuarios, no reviews separadas. Así mismo, también permite la existencia de múltiples instancias para un mismo usuario con distinto número de reviews. Dado que queremos extraer atributos del texto de un usuario, concatenaciones de texto distintas deben devolver atributos distintos, de modo que dos instancias del mismo usuario cuentan como dos casos distintos para el clasificador. Esto nos permite tener un dataset mayor que 62 instancias de usuario (una por usuario en el dataset IMDb62).
3. Generación de atributos para todas las instancias de la base de datos. La herramienta recupera las instancias de la base de datos y los generadores de atributos (en el módulo de mismo nombre) para calcular los atributos de cada una de las instancias y almacenarlos en la base de datos. Para este modo de funcionamiento se puede indicar en la base de datos que generadores de atributos queremos que estén activos. De esta forma se generarán únicamente los atributos seleccionados en la base de datos, en lugar de todos los atributos implementados.
4. Clasificación de instancias. La herramienta recupera las instancias con una configuración de atributos u otra según se especifica en la tabla NNSETUPS de la base de datos (véase ilustración 14) los separa en sets de entrenamiento y sets de validación, y alimenta estos sets a uno de los clasificadores disponibles. Los resultados se almacenan en la base de datos para su posterior análisis.

A.5. Base de datos.

En esta sección se presenta brevemente la estructura de la base de datos que se utiliza para gestionar tanto el dataset que queremos evaluar (i.e. el dataset a partir del cual creamos los IRR de usuario ya sea con los valores de SOCAL o con los de SVR) como el dataset que generamos (i.e. las instancias de texto concatenado con sus atributos) para realizar la clasificación de usuarios (i.e. predecir que método de análisis de sentimiento usar con el primer dataset a la hora de genera IRR). En la ilustración 14 podemos ver las relaciones de la base de datos. Las tablas MUSR y MREVS contienen el conjunto de los usuarios y las reviews del dataset a evaluar. Relacionado a la tabla de usuarios esta la tabla CONCATS. Esta contiene las instancias de usuarios, que se componen de la referencia al usuario y una concatenación de numrevs de reviews. Finalmente, en ATTGEN se almacenan los valores de los atributos calculados para cada instancia. MATTR es la tabla de maestro de atributos donde se almacena la descripción y la naturaleza de un tipo de atributo. Estos se dan de alta desde el módulo de generadores de atributos.

Por otro lado, desconectadas de las tablas mencionadas hasta ahora, están las tablas NNSETUPS, que contiene la configuración de atributos a utilizar en un experimento de clasificación, NNRESULTS, que almacena los resultados de ejecución de un NNSETUP, y RESULT_CLASSIFICATIONS, que almacena una simulación de cómo se calcularían los IRR de cada usuario si se utiliza el clasificador entrenado en un NNRESULT dado. Para cada usuario se almacena su MAE-pairs para SOCAL, para SVR, y el MAE-p del método elegido en clasificación (nuestro objetivo es que se elija el menor de los dos)

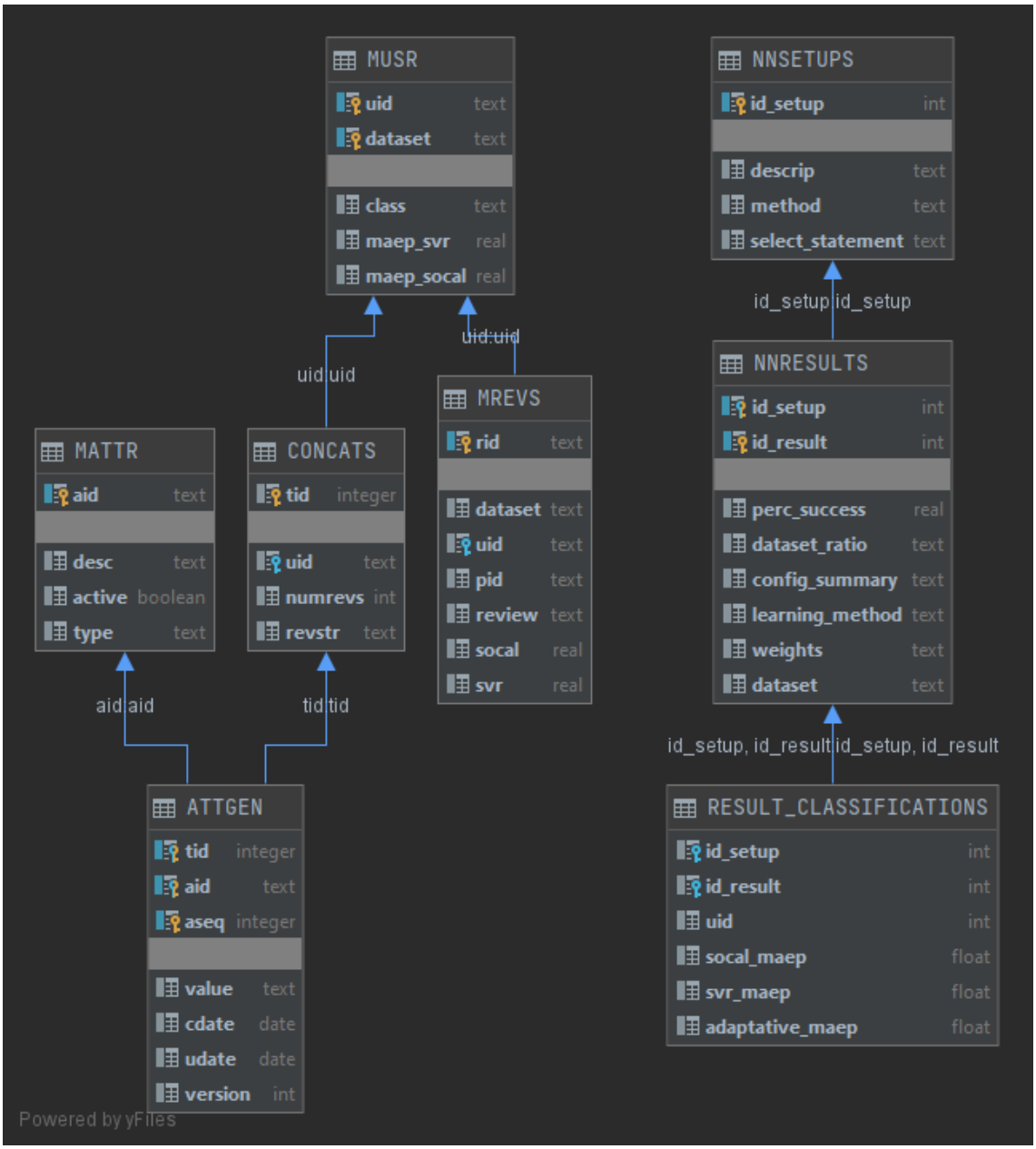


Ilustración 14: Diagrama UML de la base de datos para el almacenamiento del dataset de atributos y los resultados de clasificación.

Bibliografía

Almeida, *F.*, & Xexéo, G. (2019). Word embeddings: A survey. *arXiv preprint*.

Alswaidan, N., & Menai, *M. E.* (2020). A survey of state-of-the-art approaches for emotion recognition in text. *Knowledge and Information Systems*, 2937-2987.

Ansari, *F.*, Erol, S., & & Sihn, W. (2018). Rethinking human-machine learning in industry 4.0: how does the paradigm shift treat the role of human learning? *Procedia manufacturing*, 117-122.

AOL developers. (s.f.). Obtenido de SQLite Version 3 Overview: https://www.sqlite.org/version3.html

Bakarov, A. (2018). A Survey of Word Embeddings Evaluation Methods. *arXiv preprint*.

Balakrishnan, S., & Chopra, S. (2010). Two of a Kind or the Ratings Game? Adaptive Pairwise Preferences and Latent Factor Models. *IEEE International Conference on Data Mining*, (págs. 725-730). doi:10.1109/ICDM.2010.149

Beuscart, J. S., Mellet, K., & Trespeuch, *M.* (s.f.). Reactivity without legitimacy? Online consumer reviews in the restaurant industry. *Journal of Cultural Economy*, 458-475.

Black, D. (1948). On the rationale of group decision-making. *Journal of political economy*, 23-24.

C. Zhang, e. a. (2017). Semantic sentence embeddings for paraphrasing and text summarization,. *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, (págs. 705-709). Montreal.

Centeno, *R.*, Hermoso, *R.*, & Fasli, *M.* (2015). On the inaccuracy of numerical ratings: dealing with biased opinions in social networks. *Inf Syst Front*, 809–825. doi:https://doi.org/10.1007/s10796-014-9526-1

Conitzer, V. (2007). Eliciting single-peaked preferences using comparison queries. *Journal of Artificial Intelligence Research*, 161-91.

Conneau, A., Kruszewski, G., Lample, G., Barrault, L., & Baroni, *M.* (2018). What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *arXiv preprint*.

Cui, G. &.-K. (2012). The Effect of Online Consumer Reviews on New Product Sales. *International Journal of Electronic Commerce*, 39-58.

Dalvi, N., Kumar, *R.*, & Pang, B. (2013). Para'normal'activity: On the distribution of average ratings. *Proceedings of the International AAAI Conference on Web and Social Media*, *7.*

Duan, W., Gu, B., & Whinston, A. B. (2008). Do online reviews matter?—An empirical investigation of panel data. . *Decision support systems*, 1007-1016.

Duong, D. a.-W. (2019). Word and Sentence Embedding Tools to Measure Semantic Similarity of Gene Ontology Terms by Their Definitions. *Journal of Computational Biology*, 38-52.

Fang, H., Zhang, J., B. Y., & Zhu, Q. (2013). Towards effective online review systems in the Chinese context: A cross-cultural empirical study. *Electronic Commerce Research and Applications*, 208-220.

Fernando Enríquez, J. A.-S. (2016). An approach to the use of word embeddings in an opinion classification task,. *Expert Systems with Applications,, 66*, 1-6. Obtenido de https://doi.org/10.1016/j.eswa.2016.09.005.

Galke, L. A. (2017). Word Embeddings for Practical Information Retrieval. *INFORMATIK 2017*, 2155-2167. doi:10.18420/in2017_215

Ganu, G. *E*. (2009). Beyond the stars: improving rating predictions using review text content. *WebDB*, 1-6.

Hermoso, *R*., Centeno, *R*., & Fasli, *M*. (2014). From blurry numbers to clear preferences: A mechanism to extract reputation in social networks. *Expert Systems with Applications*, 2269-2285.

Hussein, D. *M*.-D. (2018). A survey on sentiment analysis challenges. *Journal of King Saud University - Engineering Sciences, 30*, 330-338.

Jøsang, A. &. (2009). Challenges for robust trust and reputation systems. *Proceedings of the 5th International Workshop on Security and Trust Management*, *5.* Saint Malo, France.

Kelleher, J. D., Mac Namee, B., & D'arcy, A. (2020). *Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies.* MIT press.

Khurana, D., Koli, A., Khatter, K., & Singh, S. (2017). Natural language processing: State of the art, current trends and challenges. *arXiv preprint*.

Krasnowska-Kieraś, K., & Wróblewska, A. (2019). Empirical linguistic study of sentence embeddings. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 5729-5739.

Lackermair, G. K. (2013). Importance of online product reviews from a consumer's perspective. *Advances in economics and business.*, 1-5.

Leung, C. C. (2011). A probabilistic rating inference framework for mining user preferences from reviews. *World Wide Web*, 187-215.

Luca, *M*. (2016). Reviews, reputation, and revenue: The case of Yelp. com. *Harvard Business School NOM Unit Working Paper*, 12-16.

Martens, D., Vanthienen, J., Verbeke, W., & & Baesens, B. (2011). Performance of classification models from a user perspective. *Decision Support Systems*, 782-793.

Matej Gjurković, *M*. K. (2020). PANDORA Talks: Personality and Demographics on Reddit.

Mudambi, *M*., S., & Schuff., D. (2010). Research note: What makes a helpful online review? A study of customer reviews on Amazon. com. *MIS quarterly*, 185-200.

Nadeau, D., & Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticæ Investigationes*, 3-26.

Nguyen, T. D., & al., e. (2017). Extended Named Entity Recognition API and Its Applications in Language Education. *Proceedings of ACL 2017, System Demonstrations*, (págs. 37-42).

Nils Reimers, I. G. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.

NLTK Project. (s.f.). Obtenido de NLTK 3.5 documentation: https://www.nltk.org/

Oh, K. J., Lee, D., Ko, B., & Choi, H. J. (2017). A Chatbot for Psychiatric Counseling in Mental Healthcare Service Based on Emotional Dialogue Analysis and Sentence Generation. *IEEE International Conference on Mobile Data Management (MDM)*, (págs. 371-375). Daejeon.

Pawar, A. S. (2017). Polarity Classification of Twitter Data using Sentiment Analysis. *Raghuwanshi*.

Pedregosa, *F*. a. (2011). Scikit-learn: Machine Learning in {P}ython. *Journal of Machine Learning Research, 12*, 2825--2830. Obtenido de Scikit-learn: Machine Learning in {P}ython: https://scikit-learn.org/

R. Centeno et al. (2018). From textual reviews to Individual Reputation Rankings: Leaving ratings aside solving MPC task. *ELSEVIER*.

Radhakrishnan, P. (2018). Named Entity Extraction for Knowledgebase Enhancement. *SIGIR Forum Vol. 52*, 169–170.

Ren, J., & Nickerson, J. V. (s.f.). Online review systems: How emotional language drives sales. *Twentieth Americas Conference on Information Systems.*

Sen., *E*. I. (2011). Rating: how difficult is it? *Proceedings of the fifth ACM conference on Recommender systems*, (págs. 149–156). New York. doi:https://doi.org/10.1145/2043932.2043961

Seroussi, Y., Zukerman, I., & Bohnert, *F*. (2010). A user-based approach to multi-way polarity classification. *Technical Report 2010/253*.

Seroussi, Y., Zukerman, I., & Bohnert, *F*. (2010). Collaborative Inference of Sentiments from Texts. En Y. Seroussi, I. Zukerman, & *F*. Bohnert, *User Modeling, Adaptation, and Personalization* (págs. 195-206). Berlin.

Thorsten Hennig-Thurau, K. P. (2004). Electronic word-of-mouth via consumer-opinion platforms: What motivates consumers to articulate themselves on the Internet? *Journal of Interactive Marketing, 18*(1), 38-52. doi:https://doi.org/10.1002/dir.10073

Turian, J., Ratinov, L., & Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning., (págs. 384-394).

Zhang, *R*., Gao, *M*., He, X., & Zhou, A. (2016). Learning user credibility for product ranking. *Knowledge and Information Systems, 46*(3), 679-705.