SCARR
Sistema clasificador automático de respuestas según relevancia

Rodrigo Berón Abou-Nigm
Ezequiel Jardim Godoy

Informe de Proyecto de Grado presentado al Tribunal Evaluador como requisito de graduación de la carrera Ingeniería en Computación.

Tutores
Juan José Prada - Santiago Castro

Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
28 de diciembre de 2017

Resumen

Hoy en día y en cuanto al acceso a la información mediante Internet se refiere, es de vital importancia que los datos más relevantes figuren en primer lugar al realizar cualquier tipo de consulta ya que un usuario promedio espera encontrar lo que busca en la primera página de resultados. Para facilitar la búsqueda de información es que se da lugar a sitios de preguntas y respuestas, en los que cualquier usuario puede preguntar y obtener una respuesta en poco tiempo y con una calidad confiable. Para esto, no solo es necesario que alguien responda, sino que se precisa de más personas que avalen esta respuesta sobre otras existentes, denotando su calidad y precisión. ¿Qué sucedería si se lograra prescindir de estos avales, y en su lugar se pudiera consultar un sistema automático capaz de definir con un determinado nivel de confianza esas mejores respuestas? El presente trabajo se centra en tratar de construir un modelo de aprendizaje automático que permita realizar esta tarea, según la relevancia entre las preguntas y sus respuestas. Para esto, se utilizan métodos de Aprendizaje Automático junto con herramientas de Procesamiento de Lenguaje Natural para procesar un conjunto de preguntas y respuestas. Con estos métodos se busca aprender qué características identifican la calidad de una respuesta y, en base a este criterio, poder entrenar un modelo que logre definir un orden cualitativo que le es presentado al usuario final. Tras diversas pruebas con distintos modelos de Aprendizaje Automático, como Gradient Boosting, Support Vector Machines o Multilayer Perceptrons, se logra obtener un conjunto de atributos adecuado para entrenar estos modelos. Finalmente, se desarrolla una solución que cumple con los objetivos planteados para este proyecto; aunque no se logra destacar en la clasificación individual de cada instacia —predecir correctamente la calidad de la respuesta—, sí se logra establecer el orden correcto de respuestas para una pregunta en un 76 % de las instancias evaluadas.

Índice general

Capítulo 1

Introducción

El presente capítulo define el problema que se quiere resolver, junto con los objetivos que se desean lograr. El plan de trabajo y estructura de la documentación acompañan la sección a modo de introducción al trabajo realizado.

1.1. Motivación

Hoy en día, la disponibilidad de información en línea es un hecho establecido gracias al continuo crecimiento de Internet a lo largo de las últimas dos décadas. Para muchas áreas, como la educación, es debido a este instrumento que dejaron de importar las enormes distancias tanto geográficas como culturales, las cuales se han ido estrechando, permitiendo así a personas de todo el mundo compartir su sabiduría y búsqueda de conocimiento, sin importar la lengua madre, la raza ni la nacionalidad.

A pesar de que probablemente nunca se dejará de consultar un libro en una biblioteca, preguntar a un profesor o discutir un problema con un compañero, es una realidad que una simple búsqueda en Internet ha facilitado en gran medida la resolución de muchos de los obstáculos que se presentan en el día a día. Es por esto que, a pesar de que se cuenta con un amplio abanico de recursos para despejar una duda particular, hay una gran probabilidad de que esta inquietud le haya surgido a otra persona, quien posteriormente haya contado su resolución al resto del mundo. Por esta misma razón es que en la última década se ha acentuado el crecimiento y utilización de sitios de preguntas y respuestas por parte de usuarios, también llamados *community question answering* (Q&A).

Un ejemplo muy conocido es la familia de sitios Stack Exchange, lanzada en 2009 como un simple sitio de preguntas y respuestas de programación. Este tipo de sitios no ha parado de expandirse vertiginosamente, no solo ampliando el público objetivo sino proveyendo una rica y diversa cantidad de temas a discutir —por ejemplo, literatura, matemática, trabajo, filosofía—. En ellos, los usuarios publican preguntas con una breve descripción de su problema para dar paso a otros usuarios que buscan responder con su propio conocimiento la problemática original. Luego, usuarios interesados en este problema contemplan las soluciones propuestas hasta el momento, otorgando votos a aquellas respuestas que les resultaron de mayor utilidad, y en caso de no satisfacer completamente sus necesidades les es permitido tanto contribuir con estas así como presentar nuevos enfoques. Los votos que reciben las

respuestas inciden en la relevancia de la solución en el problema particular, lo cual permite establecer un orden entre estas y también diferenciar entre soluciones útiles y soluciones irrelevantes.

Los sitios de Q&A surgen como una necesidad frente a la ineficiencia presentada por foros de preguntas y respuestas —sitios usualmente sin una estructura ni temática bien definida—, en los que usuarios realizan consultas y se generan hilos de discusión con escasa moderación hasta lograr una respuesta satisfactoria a la consulta. Estos hilos de discusión implican una pérdida de tiempo importante para el usuario, ya que este lo debe recorrer hasta dar con la respuesta adecuada, muchas veces dándose por vencido ante la cantidad de información presentada. Los sitios de Q&A —en particular Stack Exchange— están cada vez más enfocados en generar espacios donde se encuentre una respuesta de forma eficiente y clara.

Este proyecto procura seguir con esta línea de trabajo, es decir, se busca identificar las respuestas que aporten claridad en una determinada temática y a su vez descartar cualquier tipo de respuesta que no sea de utilidad.

1.2. El problema

Debido a la creciente utilización de sitios de Q&A, es de interés contar con herramientas que sean capaces de proveer automáticamente este comportamiento, ya que se reduciría la dependencia cuantitativa de personas necesarias para que sitios de este tipo funcionen correctamente. Es por esta razón que en este proyecto se propone elaborar un sistema de aprendizaje automático que permita ordenar las respuestas según su relevancia, enfocándose primariamente en la subcategoría de *English language* de Stack Exchange. Como se describe en el Capítulo 4, al comienzo del trabajo el enfoque apuntó a la subcategoría *Spanish language* de Stack Exchange, aunque tras analizar los datos disponibles para trabajar, estos resultaron ser escasos, por lo cual se apuntó a una subcategoría con mayor concurrencia de usuarios.

1.3. Objetivos del proyecto

El objetivo principal del proyecto es desarrollar una solución que lleve a cabo el sistema descripto de modo de lograr ordenar correctamente diversas respuestas según su calidad, frente a una pregunta dada. Se pretende medir objetivamente qué tan cerca está una respuesta de contestar la pregunta que la originó. Luego, se desea poder determinar un cierto orden por el que aquellas respuestas de mayor calidad sean de más interés para el usuario, discriminar aquellas de menor relevancia e incluso descartar las que sean totalmente irrelevantes.

Para lograr esto, es necesario un completo análisis de los datos disponibles, con el fin de seleccionar las mejores características de estos, para luego tomarlas y convertirlas en entradas del modelo final.

Existen diversos enfoques para atacar este tipo de problemas, por lo que se deben determinar los de mejor desempeño para los datos seleccionados. Para ello se precisa una completa investigación sobre las herramientas que se piensa utilizar para resolverlo, así como sobre los trabajos que se han realizado al día de hoy para problemas similares, o sus componentes.

Es deseable que este sistema logre obtener una buena eficacia respecto a sistemas similares estudiados. Para esto se contemplan distintos modelos de aprendizaje automático, en busca de resultados óptimos dados los datos disponibles para la resolución del problema.

Una vez desarrollada la solución, resulta interesante analizar su funcionamiento en otros sitios de Stack Exchange, como por ejemplo los relacionados a áreas como Matemática, Física, Programación o el idioma español. En particular, cuando se trata del idioma español es poco frecuente este tipo de intercambios a través de Internet, lo que resulta en un escaso análisis y una cantidad muy inferior de herramientas del área Procesamiento de Lenguaje Natural, respecto a idiomas más utilizados, como el inglés.

1.4. Plan de trabajo

La distribución de las etapas completadas a lo largo de los meses de elaboración de este trabajo se representan a continuación mediante la Tabla 1.1:

	2016				2017										
	09	10	11	12	01	02	03	04	05	06	07	08	09	10	11
Estado del arte	✓	✓	✓	✓	✓										
Construc. del *corpus*					✓	✓	✓	✓	✓						
Diseño							✓	✓	✓						
Implementación								✓	✓	✓	✓	✓	✓		
Ejecución								✓	✓	✓	✓	✓	✓	✓	
Documentación			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Tabla 1.1: Cronograma de trabajo

1.5. Organización del documento

A continuación se expone la estructura de este documento.

En el **Capítulo 2** se presenta una introducción al fundamento teórico utilizado para el desarrollo de la solución, mediante el cual se pretende mostrar un panorama básico de las herramientas utilizadas durante el trabajo.

En el **Capítulo 3** se detalla lo investigado sobre el estado del arte respecto a la temática elegida para este proyecto. Se plantean investigaciones relacionadas que

sirven de base para entender el problema, los enfoques aplicados y posibles caminos para explorar durante su desarrollo, ya sea a la hora de resolver tanto problemas similares como partes del propio proyecto.

En el **Capítulo 4** se presenta la construcción del *corpus* de preguntas y respuestas que se utilizará durante el desarrollo y la evaluación. Se describen las técnicas utilizadas para la obtención y preprocesamiento de los datos.

En el **Capítulo 5** se desarrollan los procedimientos para obtención de los atributos con los cuales entrenar los modelos de aprendizaje automático.

Una vez lista la base con la cual trabajar, en el **Capítulo 6** se propone el diseño de la solución para el problema que se quiere resolver, su implementación y resultados preliminares, que junto con los documentos anexos sirven para reportar avances y tomar decisiones.

En el **Capítulo 7** se muestran las evaluaciones experimentales ejecutadas, así como los resultados obtenidos.

En el **Capítulo 8** se detallan las conclusiones y se discuten posibles alternativas de trabajo a futuro.

Finalmente, en el **Glosario** se resumen los términos más importantes presentados en el trabajo y en la **Bibliografía** se detallan las publicaciones de los autores citados a lo largo de este.

Capítulo 2

Fundamento teórico

A lo largo de la investigación para este proyecto se consideran trabajos y herramientas relacionados con dos grandes áreas de la inteligencia artificial como lo son el Procesamiento de Lenguaje Natural y el Aprendizaje Automático. Para entender mejor el cometido de dichas áreas en este proyecto, se presenta una breve descripción de cada una así como definiciones y conceptos que se usan a lo largo de la documentación.

2.1. Procesamiento de Lenguaje Natural

> The idea of giving computers the ability to process human language is as old as the idea of computers themselves. (...) The goal of this new field is to get computers to perform useful tasks involving human language, tasks like enabling human-machine communication, improving human-human communication, or simply doing useful processing of text or speech [1].

El Procesamiento de Lenguaje Natural (PLN) es una subdisciplina de la inteligencia artificial, que consta de un conjunto de métodos y técnicas eficientes desde un punto de vista computacional para la comprensión y generación de lenguaje natural. Este proyecto se enfoca principalmente en la compresión, dejando de lado la generación de lenguaje natural [2].

Dentro de los distintos tipos de análisis se pueden encontrar:

- **Análisis lexicográfico**

  El análisis lexicográfico consiste en dividir una secuencia de caracteres, en palabras con significado propio, para posteriormente convertirla a una secuencia de términos. Estos términos se utilizan posteriormente como entrada en el análisis sintáctico.

- **Análisis sintáctico**

  El análisis sintáctico hace referencia a las relaciones de concordancia y jerarquía que tienen las palabras cuando se agrupan entre sí en forma de sintagmas, oraciones simples y oraciones compuestas.

- **Análisis semántico**

  La semántica es el estudio del significado asociado a las estructuras formales del lenguaje —sintaxis—. Así como el análisis sintáctico hace referencia a cómo las palabras se disponen en un oración, el análisis semántico se centra en determinar su significado.

2.2. Datos y componentes del análisis

Corpus

Un *corpus* es una colección de piezas de texto en formato electrónico, seleccionada de acuerdo a un criterio externo para representar, lo más posible, un lenguaje o variedad de lenguajes como una fuente de información para investigación lingüística [3].

La utilización de un *corpus* es necesaria para atacar problemas relacionados al área de PLN, puesto que tiene un rol principal a la hora de obtener los datos necesarios para resolverlos. Este conjunto de datos suele presentar un formato definido para una eficiente manipulación, además de contar con una gran cantidad de información disponible para poder enfocar y combinar distintas metodologías de resolución del problema.

```
<Texto>
 <Pregunta>
  <Titulo>
   Better phrasing for ´year we started working with [them]´?
  </Titulo>
  <Cuerpo>
   Is there a better phrasing for the title of the field, rather than ``
       YearWeStartedWorkingWith´?
  </Cuerpo>
  <Respuesta>
   Inception means the beginning of something.
  </Respuesta>
  <Respuesta>
   I am assuming you want a short concise field label, correct?
   How about one of these:
   - Worked With Since
   - Partner Since (if you consider the school systems partners)
   - Since Year
   - Year Started
  </Respuesta>
  <Respuesta>
   How about PartnershipStartDate?
  </Respuesta>
 </Pregunta>
</Texto>
```

Ejemplo 2.1: Fragmento extraído del *corpus* a utilizar

En el Ejemplo 2.1 se puede visualizar una pregunta, conformada por su título y cuerpo, y un conjunto de respuestas asociadas. Para que este conjunto sea considerado un *corpus* de datos, la cantidad de preguntas junto con sus respuestas debe ser importante y apropiada para el estudio del problema.

Es de interés notar que el volumen de información no tiene por qué ser proporcional con la calidad de la información que se puede extraer del *corpus*. Un mayor

volumen de datos será beneficioso si los tipos de datos disponibles presentan una distribución equilibrada, lo que reduce al mínimo un posible sesgo preferencial respecto a alguna característica o atributo. En caso contrario, se debe considerar una reducción del *corpus*, de modo de optimizar la calidad de la información que se pueda procesar de este.

Otro punto interesante es el tamaño asociado a un *corpus*. Muchas veces se comienza a partir de texto en crudo —sin ningún tipo de análisis previo o de acondicionamiento—, lo que sugiere la existencia de información redundante o superflua. Es necesario un manejo adecuado de esta información para acelerar los tiempos de procesamiento, dado que cuando el *corpus* es del orden de los *gigabytes* o superior, el sistema puede ver afectado su desempeño a la hora de manipular los datos.

Atributos

En particular para el tipo de problema que se quiere resolver, en el *corpus* utilizado se pueden encontrar *atributos* —también llamados *características*—, que son aquellos elementos que representan información de interés para la resolución del problema. Estos elementos pueden ser extraídos directamente del *corpus* o pueden ser combinados con distintos elementos de este.

En el Ejemplo 2.2 se puede apreciar una pregunta junto con una respuesta, y diversos elementos como el cuerpo de la pregunta o la respuesta, sus puntajes, fecha de creación o su identificador.

```
<Pregunta>
 <Titulo CantSignosPregunta=1 CantResaltadas=0 CantCursivas=0>
     Better phrasing for year we startedworking with [them] ?
 </Titulo>
 <Cuerpo CantSignosPregunta=1 CantResaltadas=0 CantCursivas=0>
   Is there a better phrasing for the title of the field , rather than
       YearWeStartedWorkingWith ?
 </Cuerpo>
 <Usuario>1156</Usuario>
 <Favoritos>0</Favoritos>
 <Puntaje>3</Puntaje>
 <FechaCreacion>2010-09-10T19:55:10.073</FechaCreacion>
 <Id>2867</Id>
 <Respuestas>
  <Respuesta SimilitudCoseno=0.0 CantSignosPregunta=0 CantResaltadas=0
      CantCursivas=0>
   <Id>2901</Id>
   <Cuerpo>
    Inception means the beginning of something.
   </Cuerpo>
   <FechaCreacion>2010-09-11T17:04:42.717</FechaCreacion>
   <Puntaje>1</Puntaje>
  </Respuesta>
 </Respuestas>
</Pregunta>
```

Ejemplo 2.2: Atributos

Segmentación

Es el proceso por el cual se divide una unidad de texto en un conjunto de elementos denominados *tokens* —en el contexto de PLN—; esta es la primera etapa por la que pasa un texto al ser analizado. Una forma sencilla de obtener los *tokens* de una oración es separando los caracteres alfabéticos que estén delimitados por un espacio. Algunos problemas en la segmentación pueden darse con nombres compuestos como *Buenos Aires*, frases como *estado del arte*, o abreviaciones como *RR. HH.*, que plantean la pregunta de si se deben considerar como uno o varios *tokens*.

Lematización

Es el proceso por el cual se toma una forma flexionada de una palabra y se obtiene su lema correspondiente, siendo el lema la forma aceptada como representante de todas las formas flexionadas de una palabra. A modo de ejemplo se observa que *canto, canta, cantamos* y *cantaron* son formas flexionadas correspondientes al lema *cantar*.

Volcado de datos

Diversas instituciones o sitios en Internet muchas veces comparten gratuitamente sus bases de datos con el fin de que los usuarios que así lo deseen puedan tener acceso a la información que manejan día a día. Este conjunto de datos suele ser llamado *dump* y en general representa un extracto de un día o mes en particular de un conjunto de datos.

Los datos normalmente se presentan sin ningún tipo de procesamiento, por lo que a la hora de generar un *corpus* es necesaria la extracción de sus atributos relevantes y el manejo adecuado de una estructura de datos eficiente para su posterior procesamiento.

2.3. Aprendizaje automático

> Ever since computers were invented, we have wondered whether they might be made to learn. If we could understand how to program them to learn (...) the impact would be dramatic [4].

A lo largo de la historia, muchos avances tecnológicos y culturales han sido posibles gracias a la detección de patrones de comportamiento en la naturaleza, la ciencia e incluso en la sociedad. El ser humano ha demostrado desde sus orígenes una particular afinidad hacia estos debido a su capacidad de explicar fenómenos desconocidos a través del modelado de situaciones presentes en situaciones comunes y corrientes, por ejemplo, conceptos amplios como la evolución [5] o el estudio del recorrido que hace una hormiga al volver a su hormiguero [6].

Sin embargo, existen problemas complejos en los que no es suficiente la sabiduría clásica para construir una herramienta que permita resolverlos de forma eficiente, y es aquí donde entran en juego los métodos de aprendizaje automático. Mediante la ayuda del poder de cómputo y de estrategias que permitan acercarse a una solución inicial a un problema dado, es posible sobrepasar el razonamiento humano en muchas ocasiones y encontrar así patrones o conocimientos necesarios para resolver el problema, lo que muchas veces escapa al entendimiento humano pero sin lugar a dudas lo hace con una eficiencia envidiable. En la actualidad, la disponibilidad de grandes volúmenes de datos permite en gran parte el estudio de muchas áreas, como el procesamiento de lenguaje natural, reconocimiento de audio e imágenes, problemas de optimización, entre otras.

En pocas palabras, la idea detrás del aprendizaje automático se basa en lograr que un modelo mejore su desempeño a través de la ejecución de una determinada acción, que es contrastada con medidas y tiempo dado. A esta ejecución se le llama *entrenamiento*, que puede tener dos tipos: **indirecto** o **directo**. El primero —que recibe el nombre de *aprendizaje no supervisado*— consiste en repetir la acción e inferir de los resultados y el contexto el camino hacia un aprendizaje correcto. El segundo —*aprendizaje supervisado*— consiste en la utilización de un atributo que contiene la clasificación correcta dentro del conjunto de atributos dados al modelo [7]. Este atributo representa lo que el modelo debe aprender a partir de las instancias de entrenamiento dadas.

Es de suma importancia definir qué es exactamente lo que se quiere que el modelo aprenda y cómo representarlo para así poder interpretar sus resultados. Por otro lado, dependiendo del tipo de problemática son aplicables distintos paradigmas y modelos, cada uno con sus ventajas y desventajas respecto al problema y a la solución buscada.

Clasificación y regresión

La primera decisión que se debe tomar a la hora de entrenar un algoritmo de aprendizaje automático supervisado es determinar si se está ante un problema de clasificación o de regresión. A continuación se describe cada tipo.

Clasificación

Dado un conjunto de elementos a aprender, cada uno perteneciente a una clase definida, la tarea de un clasificador consiste en predecir a qué clase pertenece cada elemento basándose en sus características. Si solamente existen dos clases, se dice que es un *clasificador binario*, mientras que si existen más de dos clases se dice que es un *clasificador multiclase*.

Algunos ejemplos concretos de problemas de clasificación:

- Dado un conjunto de características de un paciente —su historial médico—, determinar si el paciente padece una enfermedad específica.

- Dada la imagen de un animal, determinar si es un perro, un gato o una persona.
- Dado un correo electrónico específico, determinar si es un correo no deseado.

Regresión

Dado un conjunto de elementos, la tarea de regresión consiste en predecir un valor continuo en base a las características de los elementos. Un ejemplo concreto de regresión puede ser determinar el precio de una casa en base al tamaño, el barrio en el que está, el material de que está hecha, etc.

Word embedding

Word embedding es un conjunto de modelos de lenguaje y técnicas de aprendizaje de atributos en PLN en el que palabras o frases de un vocabulario son utilizadas para crear vectores de números reales que permitan definir un nivel de asociación entre una palabra en particular y el resto de las palabras del vocabulario, todo esto en un espacio de dimensión baja en relación al tamaño del vocabulario. Una herramienta frecuentemente utilizada para esto es Word2Vec [8].

Support vector machines

Se denomina *support vector machines* (SVM) [9] al método supervisado de clasificación cuyo entrenamiento consiste en encontrar un hiperplano que separe los vectores que representan los documentos del conjunto de datos (vectores de atributos) en dos grupos, siendo esta separación la más grande posible. Estos vectores son utilizados para buscar posibles agrupaciones de datos que puedan dar pie a la explotación de otros atributos deducibles a partir de la combinación de estas, por ejemplo mediante la herramienta WEKA [10].

Multi-layer perceptron

Un *multi-layer perceptron* (MLP) —o comúnmente conocido como *red neuronal* o *perceptrón multicapa*— es una herramienta clásica a la hora de aproximar funciones reales y discretas. Inspirada en el comportamiento del sistema nervioso, la red neuronal consiste en una interconexión de unidades simples —neuronas— donde cada una posee entradas reales y produce una salida real o discreta. Lo que ocurre dentro de dicha red suele ser considerado como una "caja negra" ya que presenta una difícil interpretación para el usuario. Si bien esta interpretación no suele ser de importancia para problemas básicos, se pueden llegar a inferir algunas características mediante el análisis de estos datos generados, que es lo que se pretende lograr en este trabajo para refinar aquellos atributos de mayor relevancia.

Una herramienta de estas características suele ser aplicada a esta clase de problemas ya que permite atacar los distintos enfoques necesarios, como buscar un

determinado orden de las respuestas, o la similitud sintáctica y semántica entre sentencias. El primer caso trata de un problema de clasificación, es decir que para un determinado conjunto de entradas se calcula una salida dentro de un conjunto finito de valores, por ejemplo, un vector de probabilidades para cada categoría. El segundo caso consiste en una regresión, lo que implica valores de salida continuos. Además, en estos casos son aceptables largos tiempos de entrenamiento, ya que se espera una rápida respuesta de ejecución una vez entrenada la red.

Árboles de decisión

Se conocen como *árboles de decisión* a las estructuras de datos que simulan la forma de un árbol, que están compuestas por nodos, donde cada uno es la unidad sobre la que se construye el árbol y puede tener 0 o más nodos hijos conectados a él. El primer elemento de un árbol es llamado *nodo raíz*, mientras que aquellos nodos terminales que no tienen ningún hijo son llamados *hojas*. Cada una de estas hojas representa un valor en la clasificación del problema que se quiere aprender, al cual se llega tras recorrer un camino desde el nodo raíz. Un método de aprendizaje conocido que utiliza dichas estructuras es ID3 [11].

La idea detrás de la utilización de estas estructuras es poder llegar a un nodo hoja que representa un valor final en la clasificación que se busca aprender. Para construir el árbol de decisión, ID3 parte de un conjunto de ejemplos conformado por series de tuplas de valores, también llamados atributos. Uno de estos atributos es binario y representa la clasificación que se busca aprender. El conjunto de ejemplos se divide en subconjuntos cada vez más pequeños a medida que se va construyendo el árbol, y se asocia cada conjunto a los posibles valores que puede tomar un determinado atributo. Como resultado final se obtiene una árbol cuyos nodos intermedios representan los posibles valores de un atributo y las hojas, el valor de la clasificación que se busca aprender.

Gradient boosting

Gradient boosting es un modelo de aprendizaje automático que se utiliza tanto en problemas de clasificación como de regresión; la idea detrás de este modelo es construir un clasificador en base a varios clasificadores débiles —de preferencia árboles de decisión—. El funcionamiento de este modelo involucra tres elementos:

- Una función de pérdida —error cuadrático—.
- Un clasificador débil —por lo general son árboles de decisión—.
- Un modelo que permita añadir un clasificador débil.

Se inicializa el modelo entrenando y evaluando las instancias definidas con lo que se obtiene una medida de error entre los valores originales y los predichos. Esta medida se utiliza en una próxima iteración con la adición de otro clasificador; el valor objetivo pasa a ser esta medida. Con esto se espera poder capturar y aprender de los errores, minimizando esta distancia iteración a iteración. Un punto importante

a tener en cuenta es la alta probabilidad de tener un sobreajuste de los datos de entrenamiento, como consecuencia de este comportamiento. Una tolerancia adecuada al nivel de error y el recorte de nodos son métodos eficientes para evitar este comportamiento.

Algunas consideraciones a tener en cuenta para este tipo de modelos son:[1]

- **Limitar la cantidad de árboles.** Con el aumento de la cantidad de árboles el modelo puede resultar muy lento. Se recomienda estudiar y limitar la cantidad máxima cuando ya deja de presentar mejorías.
- **Profundidad del árbol.** Árboles profundos suelen ser más complejos, por lo que se prefiere mantener árboles simples y con menor profundidad. Usualmente el óptimo se encuentra entre los 4 y 8 niveles.
- **Tolerancia mínima**. Se recomienda optimizar la tolerancia de acuerdo a los datos obtenidos, para poder encontrar un balance entre una convergencia rápida y una ejecución prolongada e irrelevante.

Validación cruzada

La utilización de la técnica de validación cruzada suele ser recurrente en situaciones en las que no se cuenta con una amplia cantidad de datos y por lo tanto no es factible sacrificar parte de estos datos para determinar el error del modelo. Para estimar el error se divide un conjunto de datos D en k subconjuntos disjuntos $(T_1 \dots T_k)$. Luego se procede a entrenar el modelo con el conjunto de datos $(D - T_i)$ y se valida con T_i para $i = \{1...k\}$ obteniéndose un error err_i para cada validación. Por último se tiene que el error estimado se expresa de la siguiente manera:

$$E = \frac{1}{k} \sum_{i=1}^{k} err_i$$

One hot encoding

One hot encoding (OHE) es un proceso mediante el cual atributos con valores en un dominio continuo son agrupados en clases representativas para quien modela un problema, de forma tal que permitan a un modelo de aprendizaje automático manipular de mejor manera los datos utilizados para entrenarlo y así obtener mejores resultados.

Medidas de evaluación

No siempre el porcentaje de error o el de acierto son buenas medidas, ya que no son sensibles a la cantidad de datos que se tienen. Se quiere encontrar una medida que balancee la cantidad de datos irrelevantes. Para ello, se define:

[1] https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/

- **Verdaderos positivos (VP)**

  Cantidad de instancias clasificadas correctamente como positivas.

- **Verdaderos negativos (VN)**

  Cantidad de instancias clasificadas correctamente como negativas.

- **Falsos positivos (FP)**

  Cantidad de instancias clasificadas incorrectamente como positivas.

- **Falsos negativos (FN)**

  Cantidad de instancias clasificadas incorrectamente como negativas.

Con estos conceptos, se da lugar a:

- ***Accuracy***

  Predicciones correctas sobre total de predicciones.

$$Accuracy = \frac{VP + VN}{Total} \tag{2.1}$$

- ***Precision***

  Predicciones verdaderas correctas realizadas sobre predicciones verdaderas realizadas (incluye a aquellas que incorrectamente predijo como positivo).

$$Precision = \frac{VP}{VP + FP} \tag{2.2}$$

- ***Recall***

  Predicciones verdaderas correctas realizadas sobre instancias verdaderas existentes.

$$Recall = \frac{VP}{VP + FN} \tag{2.3}$$

- **Puntaje F1**

  El puntaje F1 puede ser interpretado como un promedio ponderado de la *precision* y el *recall*; alcanza su mejor valor cuando es 1, mientras que su peor valor es 0. Se define como:

$$F1 = \frac{2VP}{2VP + FP + FN} \tag{2.4}$$

- ***Mean average precision* (MAP)**

  Al computar *precision* y *recall* en cada posición en una secuencia ordenada de documentos, se puede graficar una curva de estas dos medidas, tomando *precision* $p(r)$ en función del *recall* r. La precisión promedio muestra el valor promedio de $p(r)$ sobre el intervalo de $r \in [0, 1]$, y se puede interpretar geométricamente como el valor del área debajo de esa curva [12]. MAP se define como la media de este valor, que para un conjunto de consultas es el promedio de las puntuaciones medias de precisión para cada consulta.

- **Matriz de confusión**

  En el área del aprendizaje automático o más específicamente en problemas de clasificación, una *matriz de confusión*, también conocida como *matriz de errores*, es un diseño de tabla que permite visualizar el desempeño de un modelo de aprendizaje supervisado. Cada fila de la matriz representa las instancias que el modelo predijo como pertenecientes a una clase, mientras que cada columna representa las instancias que pertenecen a la clase. Esta matriz permite observar si el modelo está confundiendo las clases, ya que todas las instancias correctamente predichas van a encontrarse en la diagonal de la matriz. En caso contrario, de acuerdo a la distribución de los datos en la matriz se pueden sacar conclusiones, por ejemplo, dónde se agrupan las predicciones o qué clases no se logran aprender correctamente.

- **Error cuadrático medio (MSE)**

  En el área de la estadística, el *error cuadrático medio* o la *desviación cuadrática media* de un estimador mide el promedio de los errores o desviaciones elevados al cuadrado, es decir, la diferencia entre el valor estimado y el valor a estimar. Esta diferencia puede producirse de forma aleatoria o porque el estimador no cuenta con la suficiente información para producir un estimación más precisa.

- **Puntaje R2**

  Este puntaje se utiliza para medir el desempeño de un modelo de regresión; el mejor puntaje posible es 1, y puede llegar a tomar valores negativos.[2]

2.4. Stack Exchange

Stack Exchange [13] es una red de 161 comunidades al día de hoy, creadas y mantenidas por expertos y entusiastas apasionados por una temática en particular. En ella se apunta a construir un espacio de preguntas y respuestas de alta calidad, enfocadas en distintas áreas de expertiz. La familia de comunidades originalmente comenzó como un proyecto impulsado por Jeff Atwood y Joel Spolsky en el año 2008, cuando se lanzó el sitio Stack Overflow [14] —probablemente el más conocido en el ámbito ingenieril— con la finalidad de ser de utilidad para que programadores tuviesen una herramienta gratuita y en línea donde consultar y despejar sus dudas. Desde entonces no ha parado de crecer, y se estima que tiene unas 114 millones de visitas por mes.

Su objetivo principal es obtener respuestas en forma directa, sin discusiones ni desviaciones del tema. Las respuestas tienden a ser concretas, y las mejores son votadas como positivas, lo que hace que asciendan en la lista de respuestas a una pregunta.

Las mejores siempre se muestran primero, dado que se pretende facilitar la mejor opción al usuario. El foco está en atacar un problema que el usuario haya analizado,

[2] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html#sklearn.metrics.r2_score

investigado y experimentado, lo cual se considera como parte de la calidad de una pregunta. Se trata de evitar las preguntas basadas en opiniones que busquen generar discusión y se procura que siempre se centren en el tema.

Cada una de las preguntas son correspondidas con hasta un máximo de 5 etiquetas relacionadas a diversas temáticas, siempre dentro de un área común. El usuario es recompensado por su interacción con los sitios mediante la asignación —o quita— de puntos de reputación. Estos se obtienen cuando otros usuarios votan preguntas o respuestas de uno, o también editándolas. Específicamente, se obtienen +5 puntos por cada pregunta votada positivamente, +10 puntos por respuesta votada positivamente, +15 si la respuesta es aceptada, es decir, si el usuario que preguntó se siente satisfecho con la solución y la selecciona. Por último, +2 puntos si se aprueba una edición de pregunta o respuesta. A medida que se gana reputación, se desbloquean nuevos privilegios, como la habilidad de votar, comentar o editar los preguntas, respuestas o comentarios de otras personas.

Lo que se busca con este sistema de puntos es obtener las mejores respuestas a cada pregunta, apuntando siempre a la interacción en caso de que haya espacio para mejorar algún aspecto. Este es un concepto también conocido como *ludificación*, que consiste en combinar términos asociados a videojuegos en ambientes sin relación alguna con estos. Su objetivo principal es motivar a las personas mediante una mejor experiencia de usuario, por ejemplo, otorgándoles distinciones o elaborando un *ranking* según su participación [15].

Para cualquier sitio basado en Q&A, la noción de calidad de una pregunta y una respuesta es vital por muchas razones. La calidad de la información compartida promueve más calidad y mayor esfuerzo entre todos los agentes que interactúan en estos sitios, ya que aumenta la probabilidad de que alguien vuelva a consultar este sitio en caso de tener otra pregunta, sobre todo si estos sitios proporcionan un amplio abanico de temáticas para interactuar. Preguntas y respuestas de mayor calidad mejoran la reputación del sitio, dirigen más tráfico hacia él, proporcionan una mejor experiencia de usuario y fomentan que los expertos en distintas áreas busquen preguntas que pongan a prueba su nivel de conocimientos [16].

En particular, Stack Exchange promueve el concepto de reputación entre los usuarios, como modo de fomentar el compromiso del usuario con la comunidad. Esta reputación es premiada con medallas y acciones habilitadas únicamente para aquellos usuarios que tengan el nivel requerido para ello. Cada usuario comienza con una reputación de un punto, y puede tanto perder como ganar puntos en base a su comportamiento en el sitio [17], [18]. Respecto a los votos que se pueden conceder a las respuestas —tanto positivos como negativos—, se tiene una estricta reglamentación en cuanto a su utilización: de un máximo de 30 votos por día, cada usuario puede otorgar uno positivo siempre y cuando tenga al menos 15 puntos de reputación, mientras que para uno negativo se necesitan al menos 125 puntos de reputación. Este tipo de reglas limitan la cantidad de usuarios que puedan afectar una votación, ya que obliga a aportar algo de calidad antes de generar puntos de reputación.

Esto puede resultar provechoso a la hora de extraer datos, ya que poseen un cierto nivel de filtro frente a la posibilidad de que cualquier usuario que pase por primera vez y —ya sea por error o adrede— afecte la calidad real de una pregunta o una respuesta.

Alternativas a Stack Exchange

A la hora de relevar sitios interesantes de los cuales obtener información, surgieron las siguientes alternativas:

- **Quora**
  Quora es un servicio en línea donde se generan preguntas y respuestas sobre diversos temas, permitiéndosele a los usuarios colaborar en sus formulaciones. Es similar a Stack Exchange en cuanto a la cantidad de categorías presentes y a que las preguntas poseen una buena relación calidad-cantidad de respuestas. Durante octubre de 2016 se lanzó al público una versión en español pensada para la comunidad hispanohablante, y al día de hoy tiene mucha actividad. La principal desventaja de este sitio es que al momento de investigar las herramientas no se tiene una API —interfaz de programación de aplicaciones— abierta al público; si bien se han hecho muchos intentos de analizar el sitio programáticamente en búsqueda de obtener la información necesaria, la administración del sitio se ha encargado de cancelar cualquier proyecto que pueda surgir antes de que alcance un alto nivel de notoriedad, argumentando no necesitar una herramienta de este tipo aún, pese a que son conscientes de los pedidos de muchas personas [19], [20]. De todas maneras, los usuarios siguen al día de hoy intentando deducir el comportamiento del sitio en procura de acceder a algún tipo de API interna [21].

- **Ask.fm**
  Ask.fm es un sitio Q&A que está enfocado más hacia lo social que hacia lo académico, por lo que está atado a un bajo nivel de calidad de preguntas/respuestas, principalmente debido a su seriedad y estructura. Por estas razones se desistió, de antemano, de la búsqueda de herramientas para obtener acceso a sus datos.

- **Brainly**
  Brainly es una plataforma destinada a estudiantes, por lo que se puede llegar a suponer que es un ambiente académico. Sin embargo, a pesar de contar con una buena cantidad de preguntas, no se registran muchas respuestas, no suelen estar bien estructuradas ni cuentan con el nivel de calidad buscado. Por esta razón no se consideró una buena fuente de recursos para este proyecto.

- **Yahoo Answers**
  Esta es una comunidad con mucha actividad, que en su momento fue una gran fuente de recursos. Sin embargo, en su recorrido es notorio que no existe el concepto de moderación de las preguntas y respuestas, se encuentran muchos casos con comentarios impertinentes que ni siquiera se preocupan por responder la pregunta. Se consideró que un sitio así probablemente presente

contenidos incorrectos o de dificil comprensión a la hora de ser analizados —usualmente denominado *ruido*—, lo cual seguramente dificultaría la tarea de procesamiento y aprendizaje, por lo que se lo descartó también.

Capítulo 3

Estado del arte

Partiendo de los conceptos presentados en la sección anterior, en este capítulo se exponen brevemente problemas, observaciones y conclusiones de trabajos similares estudiados. Para ahondar más en estos, en el documento "Estado del Arte" anexo a este informe se describen de forma más detallada los trabajos e investigaciones que se mencionan a continuación.

> El estado del arte es una modalidad de la investigación documental que permite el estudio del conocimiento acumulado (escrito en textos) dentro de un área específica. [...] se observa que la realización de estados del arte permite la circulación de la información, genera una demanda de conocimiento y establece comparaciones con otros conocimientos paralelos a este, ofreciendo diferentes posibilidades de comprensión del problema tratado; pues brinda más de una alternativa de estudio [22].

3.1. Trabajos relacionados a SemEval 2015: tarea 3

La idea detrás del presente trabajo está basada en gran parte en una investigación planteada en la competencia SemEval[1] del año 2015 [23], en particular la tarea número tres, aunque con ciertas libertades respecto a los datos y el enfoque utilizados, procurándose una abstracción respecto de estos.

La tarea tres consistió en construir un clasificador capaz de catalogar semánticamente un conjunto de respuestas para una serie de preguntas dadas en un *corpus* de datos provistos por la entidad, obtenido de foros en Internet. Esta tarea se dividió en tres subtareas: las primeras dos con un *corpus* en inglés y la tercera con un *corpus* en árabe. En el Anexo A se presenta con mayor detalle la tarea descrita, así como los trabajos comentados a continuación.

Observando algunos resultados de esta competencia ([24], [25], [26]), se aprecia que la mayoría de los grupos participantes optan por utilizar sistemas de aprendizaje supervisados, lo cual llevó a buenos resultados.

[1] https://en.wikipedia.org/wiki/SemEval

A la hora de modelar una solución a este problema es necesario tener en claro los atributos a relevar, así como un *corpus* que estructure estos atributos y sirva de entrada al modelo que aprende la calidad de cada respuesta para una pregunta dada.

Siguiendo el enfoque anterior y guiados por los resultados de algunos competidores de SemEval 2015 [25], [27], [28], se enfoca la búsqueda de características en las siguiente áreas:

1. **Atributos relacionados a preguntas**
   - **Largo de las preguntas:**
     Categorizar las preguntas de acuerdo a una longitud (por ejemplo, preguntas cortas, medianas, largas, muy largas).
   - **Palabras:**
     Analizar las preguntas de acuerdo a las palabras encontradas, en búsqueda de temas similares.
2. **Atributos relacionados a respuestas**
   - **Análisis basado en puntuación:**
     Búsqueda de signos de puntuación específicos, como pueden ser signos de interrogación o comas.
   - **Entidades nombradas:**
     Un enlace hacia una página confiable o una cita de alguna publicación son características de una respuesta coherente y bien formada, que aumenta su calidad.
3. **Atributos sobre pares pregunta-respuesta**
   - **POS *overlap***
     Part-of-speech (POS) o *categoría gramatical* es una categorización cuya función es identificar y asignar las categorías gramaticales a las palabras (por medio de etiquetas) de un texto, como pueden ser verbos, sustantivos, adjetivos, etc. Un estándar común para categorías gramaticales es EAGLES [29], que también representa la información morfológica de cada palabra mediante un conjunto de etiquetas.
   - ***Longest common subsequences* (LCS)**
     Dados dos textos/oraciones, esta técnica se encarga de obtener la subsecuencia más larga de *tokens* en el orden que aparezcan en ambos grupos (ejemplo: “1234” y “1224533324” tiene como LCS a “1234”).
   - **Coeficiente de Jaccard**
     El coeficiente de similitud de Jaccard entre dos conjuntos de palabras se calcula dividiendo la cardinalidad de la intersección de ambos conjuntos sobre la cardinalidad de su unión. Su formulación es la siguiente:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

4. **Metadatos**
   Los *metadatos* se definen como aquellos datos que proporcionan información sobre sí mismos. Ejemplos de ellos son los siguientes campos correspondientes a los usuarios que efectúan una pregunta o una respuesta:

   - **Reputación:**
     Usuarios con buena reputación tienden a brindar buenas respuestas.
   - **Ranking respuesta:**
     Mientras más alto se encuentre en el *ranking* mejor es la calidad de respuesta.
   - **Mismo usuario que pregunta y responde:**
     Si el usuario que realiza la pregunta también la responde, puede denotar una falta de interés en la comunidad o quizás un nivel de calidad no suficiente como para satisfacer su necesidad de conocimiento. En este caso se puede inferir que de existir esta respuesta entonces seguramente sea la elegida por el usuario como respuesta final.

Dentro de los diversos enfoques encontrados al momento de relevar los modelos de aprendizaje, se destaca la utilización de ***support vector machines*** **(SVM)**, **árboles de decisión**, técnicas de ***word embedding*** y **clasificación jerárquica**.

El trabajo presentado por el grupo JAIST [27] utilizó SVM; a grandes rasgos se explica su proceder. Se obtienen los *tokens* de todas las preguntas y respuestas y se pasan a vectores utilizando Word2Vec —herramienta para modelado de *word embeddings*, como se menciona en la Sección 2.3—, donde cada palabra de la pregunta se alinea con la palabra en la respuesta cuyo vector de similitud de coseno sea mayor. En el trabajo presentado por el grupo Al-Bayan [30], se obtuvo un resultado rico en conocimiento mediante la utilización de análisis morfológico combinado con análisis semántico en árboles de decisión. En otros trabajos estudiados se utilizaron *word embeddings* en la resolución de la tarea de SemEval [25], [31].

El grupo HITSZ-ICRC propuso el método de *ensemble learning* y clasificación jerárquica para seleccionar las respuestas en cada tarea [28]. Mediante este enfoque, los resultados de la extracción de atributos demostraron que aquellos elegidos sirvieron para juzgar si una respuesta es buena o mala; aunque pocos atributos se concentraron en discriminar respuestas potencialmente útiles.

En los artículos estudiados es notorio el enfoque sintáctico que se le da a la tarea de clasificación de preguntas y respuestas, lo cual denota la falta de un estudio semántico adecuado para las relaciones entre las preguntas y sus respuestas. La mayoría de los trabajos vistos [2] correspondientes a esta tarea plantean como conclusión que tomar en cuenta atributos que midan similitud semántica podría mejorar el desempeño del modelo. Se destaca que tomar en cuenta estos atributos es una tarea no trivial, y si bien se ha utilizado en algunos proyectos [25], ha quedado como trabajo a futuro indagar más sobre componentes semánticas.

[2] Los trabajos mencionados son: [25], [27], [30], [31], [28], [26], [32].

3.2. Otros trabajos relacionados

Fuera de la evaluación de SemEval, se procura buscar información relacionada con el problema que se quiere resolver en este proyecto de grado. Para esto, y con enfoque primariamente en la falta de análisis semántico encontrado en la sección anterior, se estudian varios trabajos que proponen ideas para cubrir esta brecha.

En [26] se afirma que una herramienta utilizada para unir la información obtenida de los atributos y así obtener una relación entre un par pregunta-respuesta es una red neuronal convolutiva. En este artículo se intenta encontrar las relaciones que existen entre respuestas hechas a una pregunta, basándose en la idea intuitiva de que otras pueden servir para juzgar si la respuesta que se está analizando es potencialmente buena o no. Los autores afirman que el modelo permite capturar el enlace semántico que une un listado de respuestas, lo cual permite asignar un determinado orden a estas, además de obtener la relevancia semántica entre una pregunta y cada respuesta.

Por otro lado, en [32] se propone como objetivo centrarse en generar un sistema que posicione las mejores respuestas al principio de la lista. Si bien no se enfocan en el análisis semántico, establecen varios conceptos que resultan de interés a la hora de trabajar con un listado de respuestas a una pregunta. Se formula el problema de dos formas, utilizando aproximación *pointwise* [33] y *pairwise* [34] al problema de posicionamiento, utilizando variables binarias y evaluando con *mean average precision* (MAP).

Para evaluar el desempeño del sistema, se lo consideró como si fuese un sistema de recuperación de información en búsqueda de devolver las respuestas ordenadas según las clases de mayor orden (mejor respuesta). Como se concluyó en la Sección 3.1, se lograría mejorar ampliamente el desempeño de estos modelos si se trabajara con la semántica de los textos. Los autores también se plantean la necesidad de contemplar grandes volúmenes de datos, lo cual puede llevar a enlentecer el procesamiento de estos si no se cuenta con una arquitectura o bibliotecas adecuadas que permitan la ejecución en paralelo de los modelos.

Volviendo al análisis semántico, en [35] se propuso una continuación del trabajo presentado en SemEval 2015 de sus autores, con enfoque en este aspecto. Se plantea un modelo que contemple similitudes léxicas y semánticas, representación de vectores, el posicionamiento de una respuesta dentro del listado de respuestas de una pregunta y el posicionamiento basado en reglas. Tras diversas pruebas que contemplaron distintas combinaciones de grupos de atributos, la combinación de vectores con posicionamiento basado en reglas fue la que obtuvo mejores resultados en el conjunto de evaluación. Los autores establecieron que la utilización de un clasificador SVM demostró mejores resultados que el ganador de la competencia de la tarea tres de SemEval 2015, y además pudieron detectar determinados patrones a la hora de trabajar con este tipo de problemas: (1) aquellas respuestas "relacionadas" tienden a tener una superposición de vocabulario menor con la pregunta que aquellas respuestas "directas"; (2) las respuestas "relacionadas" a veces contienen otros términos que no están estrictamente relacionados con la pregunta; y (3) ocasionalmente, las

preguntas y las respuestas pueden estar relacionadas a través de sinónimos o lemas. Para los primeros dos puntos, a la hora de clasificar correctamente las respuestas son útiles atributos de similitud de texto. Para el último caso, se utilizan lemas y *stems* para obtener distintas formas de superficie.

En [36] se buscó encontrar los espacios léxicos existentes entre una pregunta y una respuesta de alta calidad, mediante la utilización del concepto de razonamiento análogo. Estos espacios suelen ser causados por dos factores: (1) diferencias a nivel de palabras entre una pregunta y la respuesta, y (2) respuestas poco serias o que no contribuyen a contestar adecuadamente una pregunta. Se asume la existencia de determinados enlaces latentes en los pares, como por ejemplo, las respuestas de alta calidad presentan enlaces semánticos, las poco serias presentan enlaces ruidosos, mientras que las de menor calidad o incorrectas representan enlaces inferiores. Mediante un modelo bayesiano de regresión logística, construyen una base de datos de conocimiento con estos enlaces, para que cuando se dé un par pregunta-respuesta nuevo sea posible buscar aquellos pares pregunta-respuesta similares, con el fin de obtener un vocabulario más amplio y encontrar relaciones semánticas con mayor facilidad. Logrado este conjunto, se comparan los enlaces obtenidos con los pares a resolver, y aquel enlace que represente una conexión más fuerte es el elegido como mejor respuesta. A esto los autores lo llaman *razonamiento análogo*.

En [37] se trata de enfocar en la tarea de agrupamiento de textos desde un punto de vista semántico, algo que no suele ser explotado en este tipo de modelos. Se procura organizar grandes colecciones de documentos en grupos pequeños, manejables y con un significado bien definido. Para ello, es primordial el uso de la herramienta WordNet [38], e introducen el concepto de *cadenas léxicas*.

Se plantean distintos problemas a la hora de desarrollar este tipo de modelos, como sinónimos y palabras polisémicas, o una alta dimensionalidad de atributos. Para el primer problema, se propone una medida de similitud basada en WordNet para la desambiguación de sentido de las palabras. Para el segundo punto se introducen las cadenas léxicas, con el fin de capturar el principal significado de los textos. Una *cadena léxica* se define como la secuencia de palabras relacionadas que brindan pistas importantes sobre el contenido semántico del texto. En términos de la cohesión del texto, este concepto es muy importante ya que permite analizar esta relación de manera precisa.

Los autores remarcan que la combinación de relaciones semánticas implícitas y explícitas brindó resultados positivos a la hora de la evaluación de la similitud de conceptos. A su vez, la utilización de cadenas léxicas mejora notablemente los resultados cuando se tiene una cantidad reducida de atributos disponibles en los documentos.

Por último, es importante notar que al trabajar con este tipo de sitios se tiene un sesgo en el que predomina el factor tiempo, ya que los *corpus* analizados tanto en los artículos leídos como en el que se plantea construir en la solución propuesta son datos reales que dependen de una fecha de creación y una cantidad de visitas, que están correlacionadas con la cantidad de votos finales. Es por esto que en algunos

trabajos se plantea construir un modelo de calidad de una pregunta y una respuesta que permita separar estos atributos y realizar un análisis mucho más objetivo de los datos [32].

Capítulo 4

Búsqueda y descripción del corpus

Los métodos de aprendizaje automático supervisado requieren una cantidad significativa de ejemplos etiquetados de los cuales extraer la información necesaria para entender los patrones que brinda esa información, y así poder obtener el resultado esperado. Este capítulo describe el estudio de distintas herramientas para la obtención y descripción del *corpus* que será utilizado a la hora de relevar características para el entrenamiento y evaluación del sistema final. Se explican los distintos enfoques al problema de obtención de datos, sus conclusiones y pasos posteriores.

4.1. Trabajo preliminar

A continuación se describen las distintas decisiones tomadas a la hora de buscar y obtener la información necesaria para el armado del *corpus*.

Búsqueda de datos en español

En primera instancia, el enfoque planteado para el problema de ordenamiento de preguntas en sitios Q&A se orientó a trabajar con el idioma español. Esto se debió a que no se encontraron muchos trabajos de esta índole aplicados a este idioma, lo que resulta en una motivación extra dada la posibilidad de contribuir en un área poco explorada. Dada la naturaleza del problema, se plantea la construcción de un *corpus* basado principalmente en preguntas y respuestas, para analizar las relaciones entre ellas. Para esto, se llevó a cabo una revisión de los diversos sitios comentados en el Capítulo 2; los principales fueron Spanish Stack Exchange y Quora en español.

En Spanish Stack Exchange, luego de un filtrado básico según cantidad de respuestas y de hacer un análisis cuantitativo del resultado, se pudo observar que se podía contar con apenas 2.287 preguntas y 7.117 respuestas, dentro las cuales 1.580 preguntas y 4.647 respuestas se corresponden con textos en inglés refiriéndose a dudas puntuales del idioma español. Esto deja un margen de aproximadamente 700 preguntas y 2.500 respuestas exclusivamente en español, lo que se considera escaso a la hora de atacar este problema. Para el segundo sitio —Quora en español—, debido a la falta de una API pública no se pudo acceder a la información deseada, pese a tener un buen potencial respecto a la cantidad de información disponible.

Búsqueda y extracción de datos en inglés

Debido a la escasa información en español adecuada para utilizar en esta herramienta, no fue posible extraer una buena cantidad de datos para la construcción de un *corpus* en español. Se priorizó la resolución del problema por sobre el idioma de su contenido, por lo que se optó por trabajar con un *corpus* en inglés como lo es el correspondiente a *English language* de Stack Exchange. Se describen a continuación distintas instancias en el proceso de extracción de los datos.

Stack Exchange API y CQADupStack

Tras la decisión de trabajar con un *corpus* en inglés, se procedió a relevar las herramientas para la obtención de los pares de preguntas y respuestas del sitio *English language* de Stack Exchange. Dado que Stack Exchange provee una API de acceso libre para acceder a los datos del sitio,[1] se buscaron bibliotecas que interactúen con esta API para obtener los datos de manera ordenada y estructurada, procurando que fueran lo más genéricos posibles de modo de poder reutilizar el código con otros subsitios de Stack Exchange.

Una herramienta apropiada para este fin es Py-Stack Exchange [39], que provee un uso simple, orientado a objetos y actualizado respecto a las diversas versiones de la API de Stack Exchange. Si bien los datos requeridos para utilizarla fueron simples, se presentaron dos situaciones que complejizaron su utilización.

En primer lugar, existen dos versiones de la API de Stack Exchange: una pública y una privada. Algunos datos son compartidos con una autorización previa por parte del sitio, por ejemplo, la cantidad exacta de votos positivos o negativos de una pregunta o respuesta, que va un poco más allá que el atributo *puntaje*, que se calcula como la diferencia entre ambas. La versión pública de la API establece como límite máximo unas 300 solicitudes de información por día, mientras que la privada eleva el límite a 10.000. Esta razón y la posibilidad de acceso a un mayor volumen de atributos motivaron la investigación sobre la versión privada de la API, para la cual fue necesario registrar una aplicación en el sitio Stack Apps[2] con una descripción sobre la utilización de la información obtenida.

Luego de diversas pruebas básicas, fue necesario crear consultas más complejas mediante la utilización de filtros[3] de la API. Si bien se pueden filtrar los datos según atributos básicos, por ejemplo, la fecha de creación o el puntaje mínimo, no es posible aplicar filtros personalizados, como la cantidad de respuestas de una pregunta, atributo de alto interés a la hora de relevar los datos finales. Para esto, fue necesario obtener todas las preguntas con sus respuestas y usuarios correspondientes, y posteriormente filtrarlos de acuerdo a los criterios establecidos.

[1] https://api.stackexchange.com
[2] https://stackapps.com/
[3] https://api.stackexchange.com/docs/filters

En paralelo se encontró un conjunto de datos preprocesados de referencia, a la hora de realizar investigaciones sobre comunidades Q&A llamado *CQADupStack* [40], que contiene información correspondiente a doce subforos de Stack Exchange, con sus preguntas, respuestas y usuarios asociados. Este conjunto de datos se basa en una publicación de Stack Exchange de un respaldo mensual de sus bases de datos correspondientes al mes de setiembre de 2014. Estos datos corresponden a la versión pública de la API de Stack Exchange, por lo que varios atributos de interés no se encontraron disponibles, por ejemplo, la cantidad exacta de votos positivos y negativos tanto para las preguntas como para las respuestas.

Ante la posibilidad de poder utilizar posteriormente este conjunto de datos de CQADupStack para realizar comparaciones con otros sistemas, se decidió tomar como base las preguntas y respuestas comprendidas en el conjunto y actualizar sus atributos, para tenerlos disponibles a la hora de relevar características interesantes para modelar la solución del problema de clasificación. Para llevar a cabo esta actualización, se debieron ejecutar solicitudes individuales a la API para cada una de las preguntas, sus respuestas y cada usuario, lo que elevó demasiado la cantidad de peticiones y en consecuencia resultó en una espera obligatoria —que se da cuando se excede el tope de solicitudes—. En concreto, se estimaron más de 70.000 solicitudes en el subforo de *English language* de Stack Exchange, por lo que necesariamente se debió esperar una semana para poder procesar los datos.

Para evitar otra situación como esta ante una eventual necesidad de utilizar otros subforos, se decide trabajar con el conjunto de datos provistos por CQADupStack sin modificaciones. Además, el hecho de trabajar con datos sin modificaciones permite comparar los resultados obtenidos con otros trabajos de forma directa.

Stack Exchange Data Explorer

Stack Exchange Data Explorer[4] es un subsitio donde se pueden realizar consultas SQL a la base de datos actual de Stack Exchange. A través de ella es posible visualizar o extraer los datos en CSV en tiempo real, y puede resultar útil a la hora de buscar información al día. Además, la publicación de datos correspondiente al último mes se encuentra disponible para ser descargada en el sitio Stack Exchange Data Dump.[5] Por otro lado, se encuentra disponible un respaldo de las distintas publicaciones mensuales de los datos,[6] que puede ser útil en caso de que se necesite analizar algún período en particular o evaluar la evolución de un conjunto de preguntas a lo largo del tiempo.

Otros recursos explorados

A la fecha de abril del año 2017, además de lo mencionado en la Sección 2.4, lo más cercano a una herramienta para la extracción de datos del sitio Quora es un

[4]https://data.stackexchange.com/
[5]https://archive.org/details/stackexchange
[6]https://meta.stackexchange.com/questions/224873

conjunto de datos preparados y publicados por ellos mismos,[7] con el fin de fomentar el desarrollo de herramientas de PLN desde un punto de vista semántico. Estos datos están en idioma inglés y consisten de pares de preguntas y un indicador de si son duplicadas o no. Al no poder obtener información de las respuestas relacionadas a estas preguntas, se decide dejar de lado este recurso.

4.2. Descripción del *corpus*

El conjunto de datos obtenido de CQADupStack consiste en 3 archivos compuestos por preguntas, respuestas y los usuarios por separado.

Preguntas

La información disponible para las preguntas es presentada en la Tabla 4.1.

Atributo	**Descripción**
questionid	Identificador de la pregunta
title	Título
body	Cuerpo
user	Usuario
answers_ids	Lista de identificadores de las respuestas
answers	Lista de respuestas
acceptedanswer	Respuesta aceptada
score	Puntaje
creationdate	Fecha de creación
favoritecount	Cantidad de favoritos
viewcount	Cantidad de vistas
tags	Etiquetas

Tabla 4.1: Atributos de una pregunta

En el Ejemplo 4.1 se muestra una pregunta con sus atributos en su formato original. Esta pregunta se utiliza para visualizar los atributos descriptos en este capítulo y el siguiente, y hace referencia en concreto a qué nombre poner a un campo en una tabla de base de datos de manera que sea corto y descriptivo.

[7] https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

```
{
 "questionid": 2867,
 "acceptedanswer": 2870,
 "answers": [...],
 "answers_ids": [2901, 2870, 172146],
 "body": "<p>Is there a better phrasing for the title
    of the field , rather than YearWeStartedWorkingWith
     ? </p>\n",
 "creationdate": "2010-09-10T19:55:10.073",
 "favoritecount": 0,
 "score": 3,
 "tags":  ["phrase-requests"],
 "title": "Better phrasing for year we started working
     with [ them ] ?",
 "user": {...},
 "viewcount": 189
}
```

Ejemplo 4.1: Ejemplo de pregunta

Respuestas

Los atributos de las respuestas se describen en la Tabla 4.2.

Atributo	**Descripción**
answerid	Identificador de la respuesta
body	Cuerpo
user	Usuario
score	Puntaje
creationdate	Fecha de creación

Tabla 4.2: Atributos de una respuesta

Un ejemplo de respuesta (asociada al Ejemplo 4.1) se muestra en el Ejemplo 4.2:

```
{
 "answerid": "2901"
 "body": "<p>Inception means the beginning of
    something</p>\n",
 "user": {...},
 "score": 1,
 "creationdate": "2010-09-11T17:04:42.717",
}
```

Ejemplo 4.2: Ejemplo de respuesta

Usuarios

Por último, la Tabla 4.3 describe los atributos de los usuarios.

Atributo	Descripción
userid	Identificador del usuario
age	Edad
answers	Cantidad de respuestas escritas
badges	Medallas
upvotes	Votos positivos que ha otorgado el usuario
downvotes	Votos negativos que ha otorgado el usuario
questions	Cantidad de preguntas escritas
rep	Reputación del usuario
datejoined	Fecha que se creó el usuario

Tabla 4.3: Atributos de un usuario

Un ejemplo de usuario (asociado al Ejemplo 4.1) se muestra en el Ejemplo 4.3.

```
{
  "userid": "720",
  "age": 54,
  "answers": 0,
  "badges": ['Autobiographer', 'Commentator', ...],
  "upvotes": 15,
  "downvotes": 2,
  "questions": 0,
  "rep": 760,
  "date_joined": "2010-08-08T14:35:05.090"
}
```

Ejemplo 4.3: Ejemplo de usuario

Filtrado de datos

Los datos relevados incluyen información que no es relevante para la solución del problema de ordenamiento de respuestas, por lo que se los filtra según el siguiente criterio:

1. Preguntas con al menos dos respuestas.
2. Preguntas con una respuesta aceptada por el usuario que creó la pregunta.
3. Preguntas con usuario existente.[8]

Inicialmente se parte de un conjunto de **40.860** preguntas, con **107.559** respuestas asociadas y **28.588** usuarios activos en el sitio.

Tras aplicar el primer filtro y obtener solo las preguntas con al menos dos respuestas se reduce a **27.564** la cantidad de preguntas, a **95.755** las respuestas y

[8]Cabe la posibilidad de que el usuario haya eliminado la cuenta, por lo que se podría encontrar una pregunta o respuesta sin un usuario asociado del cual extraer información.

a **23.948** la cantidad de usuarios relacionados. La elección de preguntas con una respuesta seleccionada como correcta reduce considerablemente la cantidad de información disponible, dejando un total de **18.075** preguntas, **64.381** respuestas y **16.778** usuarios.

Por último, a efectos de aprovechar los atributos disponibles de los usuarios se asegura que tanto las preguntas como las respuestas cuenten con su usuario correspondiente. Se asegura este control dado que en ocasiones puede faltar la información asociada a algún usuario en particular, no siendo posible extraer información concreta de este. Con este filtro, se reducen a **17.652** las preguntas, **62.229** las respuestas y **16.650** los usuarios. En la Tabla 4.4 se pueden visualizar a modo de resumen las cantidades de preguntas, respuestas y usuarios obtenidas tras la aplicación acumulativa de cada uno de los filtros detallados en cada fila.

Filtro	**# Preguntas**	**# Respuestas**	**# Usuarios**
Sin filtro	40.860	107.559	28.588
Al menos dos respuestas	27.564	95.755	23.948
Con respuesta aceptada	18.075	64.381	16.778
Con usuario existente	17.652	62.229	16.650

Tabla 4.4: Filtrado de datos del *corpus*

Con el fin de obtener los datos de manera simple y directa, se genera un solo archivo con un diccionario de preguntas, donde para cada una se encuentran las respuestas asociadas, así como también el usuario que creó la pregunta y los usuarios que respondieron cada una de las respuestas. Para ello se elige el formato JSON,[9] de forma de mantener la estructura jerárquica de los datos. Este formato resulta práctico y eficiente a la hora de manipularlos desde un entorno de Python, sobre todo para de abrir archivos de gran volumen de datos.

[9]http://www.json.org/

Capítulo 5

Procesamiento de atributos

Este capítulo se centra en el estudio de los atributos que serán utilizados en la construcción del sistema solución, junto con las herramientas y técnicas que permiten la manipulación y extracción de estos.

5.1. Configuración base

Una vez obtenido el *corpus* con la información necesaria para trabajar sobre el problema planteado en el Capítulo 1, se procede al armado del conjunto de datos que mejor lo representa, detallando los atributos relevantes calculados para el posterior entrenamiento.

Se utiliza el sistema operativo Linux,[1] en las distribuciones Elementary OS Freya[2] y Ubuntu 12.04.[3] El desarrollo se hace en Python,[4] dado que es un lenguaje sobre el cual se ha implementado una gran cantidad de herramientas interesantes y actuales para el análisis de PLN y aprendizaje automático.

A continuación se listan algunas de las herramientas básicas utilizadas. En secciones posteriores se mencionan otras más específicas de acuerdo al problema que se busca resolver.

Freeling

Freeling [41] es un conjunto de herramientas escritas en C++ que proveen funcionalidades de análisis de lenguaje (análisis morfológico, reconocimiento de entidades nombradas, POS-tagging, etc.) para una gran variedad de idiomas —por defecto en inglés, y opcionalmente en español—. Para el acceso a las funcionalidades, Freeling provee APIs implementadas en distintos lenguajes, en particular Python, que es la que se utiliza en este proyecto.

[1] https://www.linuxfoundation.org/
[2] https://elementary.io/
[3] http://releases.ubuntu.com/12.04/
[4] https://www.python.org/

NLTK

NLTK [42] es un conjunto de utilidades para PLN desarrollado en Python, que provee entre otras cosas herramientas para obtener los lemas, raíces, etiquetas de palabras, acceso a la base de datos léxica WordNet y otras bibliotecas externas.

Scikit-learn

Scikit-learn [43] es un módulo de Python de distribución libre que provee herramientas simples y eficientes, para el manejo de volúmenes de datos, enfocadas en la resolución de problemas de aprendizaje automático.

Jupyter Notebook

Jupyter Notebook[5] es una aplicación web de distribución libre mediante la cual se pueden crear documentos donde coexistan bloques de código Python ejecutables, y bloques de documentación —texto, imágenes, tablas, etc.—.

Pandas

Pandas [44] es una biblioteca abierta que provee estructuras y herramientas para análisis de datos con una baja curva de aprendizaje y altamente eficientes para Python.

5.2. Atributos relevados

En cuanto a las **preguntas**, analizando la Tabla 4.1, atributos como *puntaje*, *fecha de creación*, *cantidad de favoritos* y *cantidad de vistas* resultan interesantes para estudiar, dado que brindan un indicador de qué tan buena puede ser una pregunta, ya que si tiene un buen puntaje en poco tiempo puede denotar importancia e interés por parte de los usuarios, mientras que en caso contrario la pregunta puede ser poco interesante, y por lo tanto el nivel esperado de una respuesta puede llegar a bajar comparándola con una pregunta de buen nivel.

Luego, para el caso del atributo *respuesta aceptada*, en caso de utilizarse permite inferir cuál es la mejor respuesta para el usuario que realizó la pregunta, independientemente de si otra respuesta obtuvo una mayor cantidad de votos. Esto puede reflejar una diferencia de criterio entre el usuario que pregunta y el resto de los usuarios participantes del sitio, y puede darse de diversas maneras: en cierto momento se puede dar por aceptada una respuesta que con el paso del tiempo puede quedar obsoleta, o alguna respuesta mejor fundamentada puede aparecer y terminar respondiendo de mejor manera la pregunta.

[5] http://jupyter.org/

El atributo *etiqueta* representa parte del contenido temático de una pregunta y refleja el interés general del usuario que la crea, ya que busca englobar conceptos involucrados en el contenido de la pregunta. Esto se puede visualizar en la Figura 5.1. Cada pregunta tiene como máximo cinco etiquetas, con las que se pretende categorizar y facilitar la búsqueda de preguntas similares, por ejemplo, para buscar más información relacionada u obtener preguntas sin responder dentro de una categoría dada. Con esta información y técnicas de detección de contenido, como *topic models*, es viable la formulación de una relación entre pares de preguntas y respuestas a nivel de contenido.

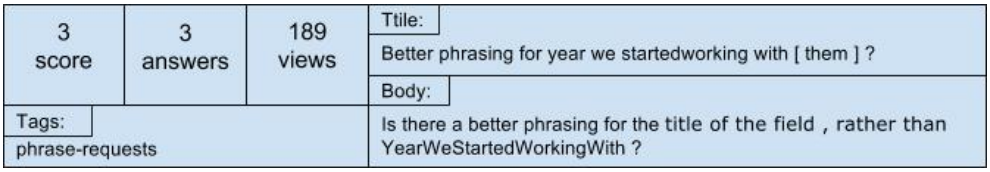

Figura 5.1: Representación de etiquetas en una pregunta

En cuanto a las **respuestas**, de acuerdo a la Tabla 4.2 atributos como *puntaje* o *fecha de creación* resultan de interés: el puntaje de una respuesta es de vital importancia para comprender qué tan relevante es para los usuarios del sitio, y va a estar estrechamente relacionado con lo que se pretende aprender en la solución. Como se mencionó anteriormente para las preguntas, la fecha de creación puede llegar a ser útil en el caso de encontrar respuestas con mayor puntaje a la respuesta seleccionada ya que se pueden encontrar nuevos enfoques que con el paso del tiempo fueron más valorados por los usuarios.

En cuanto a los **usuarios**, de la Tabla 4.3 se pueden extraer atributos como la *reputación del usuario*, que representa en gran medida qué tan bueno es desde el punto de vista de los usuarios de Stack Exchange, ya que sus acciones afectan directamente su reputación. Por otro lado, atributos como *cantidad de respuestas escritas*, *votos positivos y negativos otorgados*, o *porcentaje de selección* permiten extraer información del usuario que responde la pregunta: los primeros tres atributos pueden denotar la calidad del usuario al responder las preguntas, ya que un buen número de votos otorgados y de respuestas escritas sugieren una participación activa y responsable debido a la tendencia del sitio a mantener la calidad en las respuestas. El porcentaje de selección consiste en la fracción de respuestas del usuario que resultaron seleccionadas como mejor respuesta, sobre la cantidad total de respuestas del usuario.

Atributos calculados

Además de los atributos mencionados, se plantean a continuación otros atributos, como similitud coseno, modelado de temas o similitud sintáctica, que son atributos que no se representan de manera directa o que resultan de la combinación de distintos atributos con el objetivo de obtener más información relevante a la hora de analizar los datos.

Énfasis e información adicional de las respuestas

El cuerpo de la pregunta así como el de cada respuesta se encuentra en formato HTML, por lo tanto se utilizan expresiones regulares para extraer las etiquetas HTML de cada cuerpo y determinar si existe algún enlace externo (en particular una etiqueta <*a*>) o conceptos resaltados (con etiquetas <*b*> o <*i*>). Estos datos se guardan como atributos en un archivo XML como se muestra en el Ejemplo 5.1 :

```
<post id="2867">
<question acceptedanswer="2870" answers_ids="[u´2901´, u´2870´, u´
    172146´]"... has_link="0" has_question="2" is_bold="0" is_italic
    ="0" related="[]" .. viewcount="189">
  ....
</question>
<answers>
 <answer CosineSimilarity="0.0" answerid="2901" creationdate="
    2010-09-11T17:04:42.717" has_exclamation="0" has_link="0"
    has_question="0" is_bold="0" is_italic="0" score="1">
    ....
 </answer>
 <answer CosineSimilarity="0.241634824995" answerid="2870"
    creationdate="2010-09-10T20:10:57.107" has_exclamation="0"
    has_link="0" has_question="1" is_bold="0" is_italic="0" score="
    3">
    ....
 </answer>
 <answer CosineSimilarity="0.0" answerid="172146" creationdate="
    2014-05-22T05:13:41.260" has_exclamation="0" has_link="0"
    has_question="1" is_bold="0" is_italic="0" score="0">
    ....
 </answer>
</answers>
</post>
```

Ejemplo 5.1: Anotación de la estructura sintáctica

Relevancia del usuario en una respuesta

Cada usuario tiene medallas que le fueron otorgadas gracias a su correcta participación en el sitio. Estas medallas buscan premiar al usuario, dándole un mayor sentido de pertenencia y de responsabilidad con el sitio. Al momento de relevar las medallas, se encontraron los siguientes tipos principales:

- Medallas de preguntas.
- Medallas de respuestas.
- Medallas de participación.
- Medallas de etiquetas.

Los primeros dos tipos de medallas refieren a múltiples aspectos relacionados estrictamente con las preguntas y las respuestas, como por ejemplo la cantidad de

preguntas que se crean en un determinado rango de días o la cantidad de puntos obtenidos en una respuesta. Las medallas de participación se relacionan con la contribución del usuario al sitio a nivel personal, desde una correcta presentación, involucramiento en el desarrollo del sitio o simplemente una racha de visitas constantes. Por último, se otorgan medallas específicas para cada etiqueta frecuentemente utilizada en cada subsitio, como por ejemplo una medalla Python, Java o C# en el subsitio de *programación*. Se ignoran otras clasificaciones debido a que su contenido no es relevante para el problema de ordenamiento planteado.

Para cada categoría de medalla, se suele contar con tres escalones: bronce, plata y oro, siendo la primera la más accesible y la última la de mayor dificultad, y suelen estar relacionadas con distintas cantidades de hitos logrados, como por ejemplo obtener 10, 25 o 100 puntos en una respuesta. Existen por otra parte medallas cuyo contenido no hace referencia a una cantidad determinada, por lo que se les suele asignar uno de estos escalones de acuerdo a su importancia.

Para el relevamiento de atributos interesa tener en cuenta las medallas obtenidas por el usuario, ya que resaltan su dedicación en Stack Exchange, y si bien no existe una regla clara que especifique que un usuario con mayor cantidad de medallas responda de mejor manera una respuesta —principalmente en cuanto a calidad—, sí existe una tendencia al razonamiento de que por alguna causa este usuario posee tales medallas, lo que lleva a pensar que una respuesta puede llegar a tener mejor calidad con solo saber qué usuario respondió. Se plantea entonces la utilización de los atributos *cantidad de medallas de bronce, plata y oro*.

Similitud coseno

Se define la similitud coseno como una medida de la similitud existente entre dos vectores en un espacio que posee un producto interior con el que se evalúa el valor del coseno del ángulo comprendido entre ellos [45]. En el contexto de análisis de textos estos vectores están compuestos por sus palabras más significativas, es decir que para dos textos se busca una relación en la similitud de las palabras utilizadas como forma de expresar ideas similares. Análogamente al coseno de un ángulo, para vectores con contenido muy similar se tienen valores cercanos a uno, mientras que a mayor distancia semántica los valores se acercan a cero.

En particular, para la solución interesa estudiar los vectores de palabras de la pregunta y cada una de sus respuestas, y analizar entre sí su grado de similitud: cuanto más cercano a cero sea el ángulo compuesto entre ambos vectores, mayor es la similitud de las palabras utilizadas. En el Capítulo 6 se brinda un mayor detalle del cálculo de este atributo. Para el ejemplo descrito en el Capítulo 4, los valores de similitud coseno obtenidos se pueden apreciar en la Tabla 5.1.

Id pregunta	Id respuesta	Sim. Cos.	Mejor respuesta
2867	2901	0.0	
2867	2870	0.24	✓
2867	172146	0.0	

Tabla 5.1: Atributos de un usuario

Tanto para la pregunta como para sus respuestas se obtiene la lista de lemas que las componen y luego se comprueba si existen palabras en común. En la Tabla 5.1 se puede ver que para las respuestas con identificador 2.901 y 172.146 el valor obtenido es cero, ya que no comparten ninguna palabra en común. La mejor respuesta —con identificador 2.870— si bien tiene un valor de similitud coseno bajo, denota una cierta relación con la pregunta ya que comparte al menos una palabra.

No se espera que este atributo por sí solo presente una relación determinante entre las respuestas y una pregunta, dado que un simple caso de respuesta con el mismo texto de la pregunta resultaría en un valor de similitud coseno perfecto, pero lo que se pretende es estudiar en qué grado se producen posibles relaciones entre las palabras utilizadas a la hora de preguntar y de responder. Una limitante de este método es que no se contemplan sinónimos de las palabras, sino que se busca la presencia de palabras exactas en los vectores obtenidos. A continuación se presenta una técnica que hace foco en esta limitante, pero que ofrece alternativas que contemplen palabras similares en lugar de una búsqueda exacta.

Modelado de temas

Se describe al modelado de temas (*topic modeling* en inglés) como un conjunto de modelos que trabajan sobre un modelo estadístico para descubrir temas o categorías en un conjunto de documentos [46]. El resultado de estos modelos suele ser utilizado para resumir, visualizar, explorar y teorizar sobre un grupo de documentos. La idea principal consiste en tomar un gran volumen de datos y procesarlos, de modo de descubrir temas recurrentes en los documentos y su peso. El modelo encuentra una forma de representarlos de manera que resulte fácil y accesible la navegabilidad de la colección. Una vez obtenidos estos temas, se analizan en conjunto cada par de pregunta y sus respuestas en búsqueda de contenido relacionado procurando identificar la mayor cantidad de palabras pertenecientes a temas similares. En la sección 6.2 se pueden visualizar resultados obtenidos tras la ejecución de esta técnica.

5.3. Atributos a predecir

A partir de los atributos relevados, se plantea la predicción de dos atributos correspondientes a un par pregunta-respuesta:

- Porcentaje de puntos obtenidos por la respuesta.
- Posición de la respuesta en la lista de respuestas.

El primer atributo se plantea como la porción de puntos que una respuesta obtendría del total de puntos repartidos entre todas las respuestas a una pregunta dada. Con esto se pretende predecir qué tan buena es la respuesta en el contexto de todas las respuestas planteadas. Como consecuencia del primer punto se puede obtener un ordenamiento de todas las respuestas a la pregunta, dado que implícitamente define un orden entre las respuestas, y permite así dar una respuesta al problema planteado en la Sección 1.2.

5.4. Normalización de atributos

Luego de obtener los atributos mencionados, se procede a unificar toda la información extraída del *corpus* y de los diversos métodos aplicados.

Para esto es necesario un proceso previo de normalización de datos, puesto que los dominios sobre los que se representan los distintos atributos son dispares entre sí, lo que puede ser problemático a la hora de aplicar modelos de aprendizaje automático. Un ejemplo básico de este problema es que la mayoría de los clasificadores automáticos utilizan la distancia euclideana para calcular la distancia entre dos puntos, y si uno de los atributos tiene un amplio rango de valores, entonces esta distancia va a estar condicionada principalmente por este atributo, lo cual no es deseable. En particular, modelos como SVM suelen requerir que la media y la desviación estándar estén normalizadas. Para ello, se puede utilizar un método de la biblioteca Scikit-Learn llamado StandardScaler, que permite calcular los valores estandarizados mediante la siguiente fórmula:

$$x' = \frac{x - \text{mean}(x)}{\text{std\_dev}(x)} \tag{5.1}$$

donde $mean(x)$ refiere a la media del valor y $std\_dev(x)$ refiere a la desviación estándar del valor.

5.5. Unificación y anotación del conjunto final de datos

El último paso en el armado del conjunto de datos final es la unificación de todos los atributos que lo comprenden, de forma tal de facilitar el procesamiento posterior de los métodos de aprendizaje automático elegidos para la resolución del problema planteado.

Se genera una estructura de datos formada por cada uno de los atributos obtenidos, calculados y normalizados que serán utilizados para aprender los valores objetivos especificados en la Sección 5.3. Poder establecer qué atributos utilizar en el procesamiento permite comparar distintas agrupaciones de atributos en búsqueda de la mejor configuración.

#	**q_id**	**ans_id**	**q_score**	**q_answers**	**...**	**ans_percentage_score**
0	103.02	103.04	0.04	0.03	...	15.0 %
1	103.03	103.08	0.04	0.03	...	85.0 %
2	113.54	113.55	0.030	0.029	...	35.0 %
3	18.54	18.57	0.07	0.14	...	16.0 %
...	...	...	...	...	...	...

Tabla 5.2: Anotación del conjunto de datos relevados

En la Tabla 5.2 se puede ver una estructura de objetos compuestos por los identificadores de pregunta y respuesta; un conjunto de atributos con su valor y el valor objetivo esperado —*ans_percentage_score*—.

Finalmente, los atributos expuestos en esta sección se resumen en la Tabla 5.3, en la que se muestra el conjunto de atributos final con los cuales se procesarán las respuestas a las preguntas.

Atributos de cantidad	
q_named_entity_count	ans_user_bronze_amount
q_noun_count	ans_user_down_votes_count
q_pronoun_count	ans_user_gold_amount
q_sentences_count	ans_user_silver_amount
q_uppercase_word_count	ans_user_up_votes_count
q_user_bronze_amount	ans_verb_count
q_user_gold_amount	ans_word_count
q_user_silver_amount	q_score
q_verb_count	user_ans_selected_prob
q_view_count	ans_bold_marks
q_word_count	ans_exclamation_marks
q_answer_count	ans_href_marks
q_favorite_count	ans_italic_marks
ans_named_entity_count	ans_question_marks
ans_noun_count	q_bold_marks
ans_pronoun_count	q_exclamation_marks
ans_sentences_count	q_href_marks
ans_uppercase_word_count	q_italic_marks
ans_user_answer_count	q_question_marks
Atributos de longitud	
ans_longest_word_length	q_longest_word_length
ans_sentence_average_length	q_sentence_average_length
ans_word_average_length	q_word_average_length
Atributos de booleanos	
q_accepted_answer	
Atributos de distancia	
hellinger	jaccard
kullback_leibler	q_ans_cos_sim
Atributos de reputación	
q_user_reputation	ans_user_reputation
Atributos de fecha	
q_creation_date_days	ans_creation_date_days
Atributo	
ans_quality	
Atributos de calidad entre pregunta y respuesta	
ans/q_named_entity_count	ans/q_noun_count
ans/q_pronoun_count	ans/q_sentence_average_length
ans/q_sentences_count	ans/q_uppercase_word_count
ans/q_verb_count	ans/q_word_average_length
ans/q_word_count	

Tabla 5.3: Conjunto final de atributos a utilizar

Capítulo 6

Desarrollo de la solución

Este capítulo se centra en el diseño de la arquitectura y la toma de decisiones para el desarrollo de la solución. En la Sección 6.1 se describe el diseño propuesto para la solución y en la Sección 6.2 se hace una breve descripción de los pasos más relevantes tomados a la hora de desarrollarla, junto con resultados parciales. Por pruebas y un mayor detalle de los resultados obtenidos en los pasos intermedios, ver los Anexos C, D, y E.

6.1. Diseño

Para problemas relacionados con datos en crudo, es necesario realizar un pre y post procesamiento de estos, antes y después de haberlos adaptado a la realidad con la cual se trabajará. En particular para esta solución, se realiza una etapa de filtrado y limpieza de datos, posteriormente se procede a extraer los distintos atributos descritos en la Sección 5 para armar un *corpus* con el cual desarrollar y entrenar modelos de aprendizaje automático. En la Figura 6.1 se puede ver reflejada esta idea.

Dada la naturaleza iterativa del desarrollo de este tipo de soluciones, se opta por seguir un diseño modularizable y parametrizable, el cual permite completar ciclos de extracción de atributos y entrenamiento de modelos, realizar pruebas y sacar conclusiones para investigar alternativas tanto a nivel de atributos como de los modelos utilizados.

Se utiliza Jupyter Notebook dada su facilidad para modularizar y paralelizar las tareas de desarrollo. Una vez extraída la información del conjunto de datos que se analizarán y persistida en un formato que facilite su manejo, se avanza en paralelo hacia el objetivo marcado mediante la generación de los atributos. Incluso sin terminar de desarrollar todos los atributos pautados, se puede comenzar a probar modelos. A medida que nuevos atributos se generan, se los incluye en el clasificador y se itera el proceso.

La posibilidad de ejecutar bloques de código independientes por cada módulo agiliza la obtención del resultado deseado, puesto que ahorra tiempo significativo comparado a cuando se tiene que ejecutar en cada momento del desarrollo el código en su totalidad.

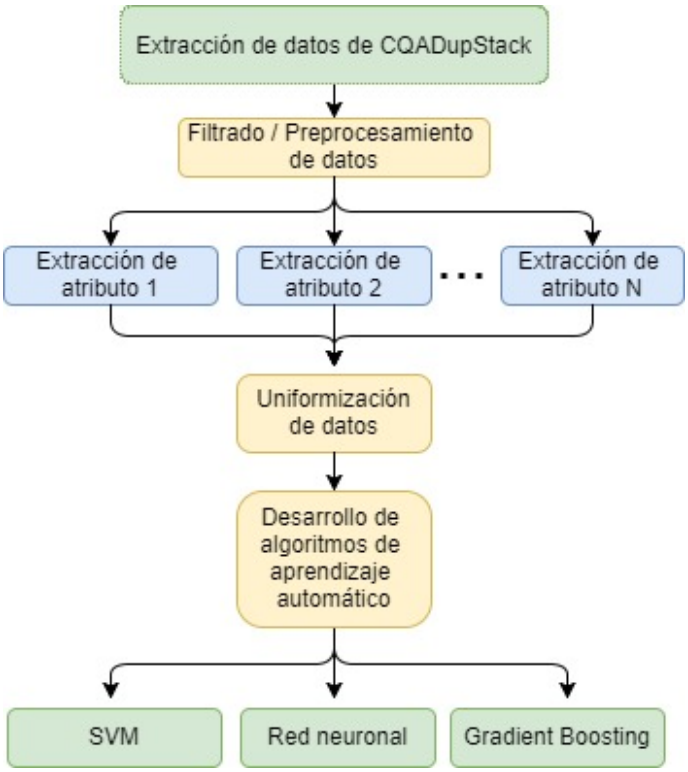

Figura 6.1: Arquitectura de la solución

6.2. Implementación

Tras definir la arquitectura de la solución, se comienza a trabajar en su desarrollo. Uno de los objetivos principales es proveer la capacidad de permutar distintas combinaciones de atributos con el mínimo esfuerzo. Para lograr esto, el foco principal consiste en desarrollar módulos independientes que traten los atributos seleccionados para generar nuevos atributos a partir tanto de combinaciones como de la aplicación de distintas técnicas de procesamiento, como se define en el Capítulo 5.

Una vez que todos los atributos disponibles son procesados, el conjunto final de datos es persistido y queda a disposición de la solución para ser filtrado por un módulo de configuración. Luego se procede a comprobar su utilidad mediante modelos de aprendizaje automático como SVM, redes neuronales y *gradient boosting*.

A continuación se describe el flujo principal de tratamiento de datos y posterior entrenamiento. Los títulos en negrita indican la etapa del flujo, mientras que con títulos en cursiva se agrupan características específicas, por ejemplo, el procesamiento de distintos atributos. En los Anexos C y D se pueden encontrar flujos paralelos de desarrollo con los que se pudo realizar pruebas y validar ideas, buscando mejorar los resultados obtenidos de modo de definir un conjunto final óptimo de atributos sobre el cual poder ejecutar las pruebas finales de la solución y comparar los resultados con otros sistemas.

Armado y filtrado del *corpus*

La etapa inicial se centra en el armado de un conjunto base de atributos, tanto sean originales, compuestos o nuevos atributos creados a partir de otros existentes. En este módulo se parte del conjunto de datos obtenidos de CQADupstack [40], y mediante la utilización de la biblioteca Pandas [44] se carga en memoria la estructura completa de los datos. Una vez realizado esto, se filtran los datos con el criterio detallado en la Sección 4.2 y se persisten en formato JSON. Cada uno de los objetos finales está compuesto por una pregunta, el usuario que crea la pregunta y una lista de respuestas junto con el usuario que las crea.

Extracción de atributos

Luego del preprocesamiento inicial de los datos, se procede a extraer los atributos que son utilizados a la hora de estudiar las posibles relaciones entre una pregunta y sus respuestas.

Similitud coseno

En la Sección 5.2 se describe el atributo *similitud coseno* como una medida para ver qué tan similares son dos vectores de palabras. En este módulo se toma cada par pregunta-respuesta existente en el conjunto de datos extraído y se calcula su valor de similitud coseno. Tanto los contenidos de las preguntas como los de cada una de sus respuestas son formateados a cadenas de texto sin etiquetas HTML como negritas, cursivas, íconos o enlaces. Una vez procesados los cuerpos, se extraen los lemas de cada palabra y se filtran las palabras poco frecuentes. Luego, se genera una matriz *tf-idf*, que lleva la cuenta de la frecuencia de ocurrencia de los términos en la lista de documentos, y con esta matriz se calcula el valor deseado mediante el método *cosine_similarity* provisto por la biblioteca Scikit-learn.

Una vez calculados todos los atributos, se genera un archivo compuesto por los identificadores de las preguntas y sus respuestas, junto con el valor de similitud coseno, y se guarda para una posterior unificación de atributos.

Metadata

Este módulo se centra en obtener atributos de metadatos. De Stack Exchange[1] [2] se extrae directamente un listado de las medallas existentes en el sitio junto con sus descripciones. Además, se extraen de una consulta SQL[3] las cantidades exactas de medallas otorgadas en total y sus tipos. Dado que se trabaja con datos de hasta setiembre de 2014, como se menciona en la Sección 4.1, se toman las medallas del usuario hasta dicha fecha.

[1] https://stackoverflow.com/help/badges
[2] http://stackoverflow.com/help/badges?tab=tags
[3] https://data.StackExchange.com/english/query/new

Además, cabe mencionar que las medallas que posee un usuario se calculan en el momento, con base en la fecha de creación de la pregunta o respuesta.

Recordar que en la Sección 4.2, al aplicar el filtro de preguntas con respuesta aceptada se obtienen 17.652 preguntas, dentro de las cuales se calcula que para un total de 8.406 preguntas la respuesta seleccionada como aceptada fue creada por el usuario con mayor cantidad de medallas. A su vez para un total de 8.342 preguntas, esta respuesta fue creada por el usuario con mayor reputación entre los usuarios de las respuestas. Estos números presentan aproximadamente un 50 % de las preguntas, por lo que resulta interesante el estudio de estos atributos a la hora de predecir una respuesta de buena calidad.

Como metadatos se plantea obtener los siguientes campos:

- ID de la pregunta y de la respuesta.
- Cantidad de medallas de bronce al momento de preguntar y responder.
- Cantidad de medallas de plata al momento de preguntar y responder.
- Cantidad de medallas de oro al momento de preguntar y responder.

Topic modeling

Tras lo presentado en la Sección 5.2, el objetivo principal de esta técnica es poder extraer información representativa de la temática de las preguntas y sus respuestas, y buscar una relación entre los temas que presenten en común. Para esto se utiliza una biblioteca Python llamada Gensim [47] —cuyo foco principal es el modelado de temas, indexación de documentos y obtención de similitudes a partir de grandes volúmenes de texto— y en particular el modelo Latent Dirichlet Allocation (LDA).

Los modelos LDA [48] son modelos generativos —en un contexto de aprendizaje automático— que permiten que conjuntos de observaciones puedan ser explicados por grupos no observados, buscando encontrar relaciones entre datos similares. Por ejemplo, si las observaciones son palabras en documentos, presupone que cada documento es una mezcla de un pequeño número de categorías —también denominados *temas*— y la aparición de cada palabra en un documento se debe a una de las categorías a las que el documento pertenece. Un modelo se dice generativo [49] cuando aprende a partir de cómo se generan las observaciones, dicho de otra manera, a partir de la probabilidad conjunta $P(C, D)$ (donde D es el documento y C la clase), y realiza la predicción maximizando la probabilidad condicional $P(C|D)$ que es calculada utilizando el teorema de Bayes.

A continuación se resume lo realizado en este módulo:

1. Extracción del vocabulario para entrenar un modelo LDA.
2. Limpieza de etiquetas HTML de los cuerpos de preguntas y respuestas y obtención de lemas y *stems*.

3. Transformación de datos para que sean legibles por Gensim.
4. Entrenamiento del modelo LDA con este vocabulario y obtención de los temas.
5. Visualización de los temas obtenidos.
6. Definición de métricas y medidas de distancia entre temas.
7. Visualización del grafo de distancias.
8. Extracción de atributos de distancia entre los temas de cada pregunta y sus respuestas.

Un mayor detalle en cuanto al funcionamiento de esta técnica, la generación del modelo LDA, la visualización de los temas obtenidos y un ejemplo ilustrativo se puede encontrar en la tercera parte del Anexo C.

Medidas de distancias entre vectores de temas

Se plantea la utilización de una medida de distancia llamada *distancia de Hellinger*, la *entropía relativa* —también llamada *divergencia de Kullback–Leibler*— y la *distancia de Jaccard*.

La distancia de Hellinger es utilizada para calcular la distancia entre dos distribuciones de probabilidad. El dominio de salida es un número real entre [0,1] —mientras más cercana a 0, más similar es—. Por otro lado, la divergencia de Kullback-Leibler es una medida que se define como la divergencia entre dos distribuciones de probabilidad P y Q. En aplicaciones, P suele representar una distribución correcta de la información —la temática o palabras clave de una pregunta—, mientras que Q representa una aproximación de P —la temática o palabras clave de una respuesta—. Cuanto más cercanas sean estas probabilidades se obtendrá un valor más cercano a 0, por lo que las características semánticas extraídas tanto de la pregunta como de la respuesta tienden a ser iguales. La distancia de Jaccard entre dos conjuntos de palabras consiste en la cantidad de coincidencias que se tienen sobre la unión de los dos conjuntos, como se detalla en la Sección 3.1.

Similitud sintáctica

Se utilizan estructuras XML para almacenar tanto atributos relevados directamente de Stack Exchange —reputación de un usuario, cantidad de votos de una respuesta, entre otros— como atributos que requieren un mayor procesamiento (similitud semántica, árbol sintáctico de una respuesta o pregunta, etc.). Por cada pregunta de Stack Exchange se genera un archivo XML que contiene los atributos relevados y procesados como se muestra en el Ejemplo 6.1.

La principal razón por la que se utilizan estructuras XML es que estas permiten representar el árbol sintáctico del cuerpo de cada pregunta y sus respectivas respuestas. Esto permite obtener y calcular atributos sintácticos de forma rápida y eficiente. En el Ejemplo 6.1 se ilustra un ejemplo de la estructura sintáctica para

parte del cuerpo de la pregunta *Better phrasing for 'year we started working with [them]'?*.

```
<S>
 <adv><word lem="well" tag="RBR" wd="Better" /></adv>
 <vb-chunk><word lem="phrase" tag="VBG" wd="phrasing" /></vb-chunk>
 <prt-mix><word lem="for" tag="IN" wd="for" /></prt-mix>
 <n-chunk><word lem="year" tag="NN" wd="year" /></n-chunk>
 <sn-chunk><word lem="we" tag="PRP" wd="we" /></sn-chunk>
 <vb-chunk><word lem="start" tag="VBD" wd="started" /></vb-chunk>
 <vb-chunk><word lem="work" tag="VBG" wd="working" /></vb-chunk>
 <prt-sp><word lem="with" tag="IN" wd="with" /></prt-sp>
 <inc-mk><word lem="[" tag="Fca" wd="[" /></inc-mk>
 <sn-chunk><word lem="them" tag="PRP" wd="them" /></sn-chunk>
 <inc-mk><word lem="]" tag="Fct" wd="]" /></inc-mk>
 <in-brk><word lem="?" tag="Fit" wd="?" /></in-brk>
</S>
```

Ejemplo 6.1: Anotación de la estructura sintáctica de una pregunta

Estos archivos se utilizan para relevar tanto atributos sintácticos como atributos estadísticos, algunos de los cuales se detallan a continuación:

- **Atributos sintácticos**
  - *q_named_entity_count* — Cantidad de entidades nombradas en la pregunta.
  - *q_noun_count* — Cantidad de sustantivos en la pregunta.
  - *q_pronoun_count* — Cantidad de pronombres en la pregunta.
  - *ans_named_entity_count* — Cantidad de entidades nombradas en la respuesta.
  - *ans_noun_count* — Cantidad de sustantivos en la respuesta.
  - *ans_pronoun_count* — Cantidad de pronombres en la respuesta.
- **Atributos estadísticos**
  - *q_longest_word_length* — El tamaño de la palabra más larga en la pregunta.
  - *q_sentence_average_length* — El promedio del largo de sentencia en la pregunta.
  - *q_uppercase_word_count* — Cantidad de palabras que empiezan con letra mayúscula en la pregunta.
  - *ans_longest_word_length* — El tamaño de la palabra más larga en la respuesta.
  - *ans_sentence_average_length* — El promedio del largo de sentencia en la respuesta.
  - *ans_uppercase_word_count* — Cantidad de palabras que empiezan con letra mayúscula en la respuesta.

Se genera una lista de diccionarios; cada uno contiene un *q_id*, un *answer_id* —pares de identificadores pregunta-respuesta respectivamente— y atributos que pueden pertenecer a la pregunta o a la respuesta. La lista se utiliza posteriormente en la unificación de atributos; un ejemplo de esta se observa en el Ejemplo 6.2.

```
[{
 "q_id": 2867,
 "answer_id": 3613,
 "kullback_leibler" : 13.1736,
 "hellinger": 0.766317,
 "jaccard": 0.869999,
 ...
 "ans/q_word_count": 0.14,
 "ans/q_word_average_length": 0.88,
 "ans/q_sentences_count": 0.13
 ...
},
{
 "q_id": "4711",
 "answer_id": "4715",
 ...
}]
```

Ejemplo 6.2: Cálculo de atributos XML

Atributos generales

Se definen los atributos generales como aquellos que se obtienen de forma directa de los datos extraídos de Stack Exchange —o con algún cálculo mínimo—. Estos atributos se almacenan en un archivo con formato JSON, y están compuestos por:

- ID de la pregunta y de la respuesta.
- Puntaje de la pregunta y de la respuesta.
- Fecha de creación de la pregunta y de la respuesta.
- Cantidad de usuarios que vieron la pregunta.
- Existencia de una respuesta aceptada.
- Cantidad de respuestas a la pregunta.
- Existencia de etiquetas HTML en la pregunta y en la respuesta.
- Porcentaje de puntos que tiene la respuesta respecto a la cantidad total de puntos de todas las respuestas.
- Reputación del usuario de la pregunta y del usuario de la respuesta.
- Cantidad de respuestas, votos positivos y negativos del usuario de la respuesta.

El *valor objetivo para el modelo de aprendizaje automático* se define como el **porcentaje de puntos que tiene cada respuesta, dada la suma de todos**

los puntajes de las respuestas. Esto pretende reflejar la proporción de puntos totales repartidos entre todas las respuestas.

Dado que el puntaje de una respuesta se calcula[4] sumando los *upvotes* (votos positivos) y los *downvotes* (votos negativos), pueden darse valores positivos, negativos o iguales a 0. Esto hace que el puntaje total pueda llegar a ser menor o igual a 0, por consiguiente, el cálculo del porcentaje que tiene una respuesta no es tan directo como se pensó en un principio:

$$\text{porcentaje} = \frac{\text{puntaje de respuesta}}{\text{puntaje total}}$$

Para mitigar este problema se calcula el puntaje total negativo como la suma de todos los puntajes negativos, y el positivo como la suma de todos los puntajes positivos. Luego, cuando se quiere calcular el porcentaje de puntos de una respuesta, si la suma es negativa se divide entre la cantidad total negativa, y análogamente si es positiva. De esta forma, se busca ponderar la calidad de una respuesta: si es buena, qué tan buena es y, a su vez, si es mala, qué tan mala es.

En la Tabla 6.1 se puede visualizar un ejemplo para remarcar el concepto descrito.

N.° Respuesta	**Puntaje**	**Porcentaje**
1	9	90 %
2	1	10 %
3	0	0 %
4	-3	-75 %
5	-1	-25 %

Tabla 6.1: Porcentaje de puntos de respuestas respecto a su pregunta

Luego de procesar todos los datos y sin haber aplicado *one hot encoding* se cuenta con un total de **63** atributos, mientras que luego de aplicarlo se tiene un total de **149** atributos. Ver una descripción más detallada de los atributos en el Anexo B.

Atributos de calidad y cantidad en una respuesta

Existen atributos relevados del cuerpo de una respuesta que potencialmente reflejan su calidad, como por ejemplo si una respuesta cuenta con enlaces, probablemente sea una referencia a materiales que el usuario usó para generar la respuesta. Por otro lado existen atributos que no reflejan calidad en una respuesta, por ejemplo, los signos de interrogación, que probablemente sea una duda sobre la pregunta, más que una respuesta para la pregunta en sí.

Con el objetivo de generar un atributo que refleje la calidad de una respuesta para una determinada pregunta se investiga el dominio de los atributos que por sí solos no reflejan características relevantes de la pregunta o las respuestas, con el objetivo de definir una función que genere un nuevo atributo que sí lo haga. El primer

[4]https://meta.stackexchange.com/questions/229255/what-is-the-score-of-a-post

paso para crear la función es estudiar y entender el dominio de estos atributos, los cuales se presentan en la Tabla 6.2.

	mean	**std**	**max**	**min**
ans_bold_marks	0.77	2.00	108.0	0.0
ans_exclamation_marks	0.12	0.66	36.0	0.0
ans_href_marks	0.61	1.50	141.0	0.0
ans_italic_marks	0.05	1.24	207.0	0.0
ans_question_marks	0.42	1.22	64.0	0.0

Tabla 6.2: Media, valor máximo, mínimo y desviación estándar para los atributos

En esta tabla se observa que el valor medio de cada atributo —*mean*— se encuentra más próximo al mínimo de su dominio, por lo tanto es esperable que la cantidad de ejemplos que tengan valores altos para estos atributos sean una minoría dentro del conjunto de datos. Para tener una mejor idea del dominio de cada atributo se realiza un análisis de la cantidad de ejemplos por valor para cada atributo. Como se observa en la Tabla 6.2 algunos atributos alcanzan hasta 200 valores diferentes, por lo cual hacer este análisis para cada uno de estos 200 valores no es algo factible y en su lugar se opta por dividir el dominio de cada atributo en cinco clases y se calcula la cantidad de ejemplos que se encuentran en cada una para cada atributo. La cantidad de clases que se utilizan se encuentra fuertemente vinculada a la información que se busca reflejar. Una primera idea podría ser dividir el dominio de cada atributo en tres clases, dependiendo de si el valor del atributo se encuentra debajo, cerca o por encima de la media. Dado que se busca un nivel de granularidad mayor respecto al dominio del atributo, se agregan dos clases más que permitan reflejar la distancia —por encima o por debajo— a que se encuentra el valor con respecto a la media, quedando así cinco clases. Estas clases se pueden visualizar en la Tabla 6.3.

	Clase 1	**Clase 2**	**Clase 3**	**Clase 4**	**Clase 5**
ans_bold_marks	41615	20572	34	7	1
ans_exclamation_marks	57942	4246	32	5	4
ans_href_marks	42891	19333	4	0	1
ans_italic_marks	60566	1659	2	0	2
ans_question_marks	48564	13630	33	1	1

Tabla 6.3: Cantidad de ejemplos por clase

Luego, se define el conjunto *atr_calidad* como la agrupación de los primeros cuatro atributos de la Tabla 6.3, y se define *atr_no_calidad* como los atributos que no reflejan calidad, en este caso representado únicamente por el atributo *ans_question_marks*.

Una instancia de respuesta tendrá cada uno de estos atributos enmarcados en una de las clases definidas en la Tabla 6.3, por lo que se define la calidad de una respuesta como:

$$calidad = (\sum_{atr\ \epsilon\ atr\_calidad} clase_i \quad - \sum_{atr\ \epsilon\ atr\_no\_calidad} clase_i)$$

donde $clase_i$ toma valores entre 1 y 5, dependiendo de la clase a que pertenezca cada uno de los atributos de la instancia.

El dominio de este nuevo atributo —que se denomina *quality*— se muestra en la Tabla 6.4.

	mean	**std**	**max**	**min**
quality	0.52	0.82	7	-2

Tabla 6.4: Media, desviación estándar, máxima y mínima del atributo de calidad

Atributos de cantidad

Un proceso similar al anterior se hace para los atributos de cantidad (*ans_named-_entity_count, ans_noun_count, ans_pronoun_count*, etc.) solo que esta vez se calcula la relación entre el atributo de cantidad perteneciente a la pregunta y el perteneciente a la respuesta. Para entender esta idea se muestra un ejemplo de cómo se calcula la cantidad de contenido de la respuesta respecto a la pregunta:

$$ans/q\_sentence\_average\_length = \frac{ans\_sentence\_average\_length}{q\_sentence\_average\_length}$$

Es deseable y esperable que dentro de la respuesta a una pregunta se encuentren oraciones más largas que en la pregunta, las cuales intentan explicar un determinado concepto. Se procede de forma análoga con el resto de los atributos de cantidad (*named_entity_count, pronoun_count, sentence_average_length, uppercase_word_count, verb_count, word_average_length, word_count*) donde para cada uno se genera un nuevo atributo que refleja la relación de cantidad entre la respuesta y la pregunta, denotado con el prefijo *ans/q_*.

Unificación de atributos

Para cada uno de los puntos anteriores se guardan tablas de datos creadas con la biblioteca Pandas. Cada una de sus filas representa una respuesta a una pregunta y sus atributos relacionados. Dado que estas tablas comparten estos identificadores, se cruzan entre ellas para generar un único archivo con las respuestas y todos sus atributos. Una vez creado el conjunto unificado, se verifica la falta de atributos para los pares pregunta-respuesta, donde se resuelve esta falta en cada caso acorde con el tipo de atributo.

Los algoritmos provistos por Scikit-learn no suelen manejar correctamente la falta de información en los datos de entrada, por lo que valores no válidos como

lo son **NaN** (valores no numéricos) o ∞ (valores infinitos) deben ser corregidos, usualmente utilizando métodos que permitan asignarles valores sin que introduzcan cambios en la distribución de los datos. Luego de realizar esto, para todos los atributos —menos los correspondientes a los identificadores de las preguntas y el atributo objetivo ***ans_percentage_score***— se normalizan los datos de acuerdo a lo presentado en la Sección 5.4.

Análisis de atributo objetivo

Como se puede ver en la Tabla 6.1, el objetivo principal de la solución desarrollada es poder aprender qué porción de los puntos que se reparten a lo largo de las respuestas de una pregunta corresponde a cada una de las respuestas.

Debido a que estos valores pueden pertenecer al rango [-100, 100], en caso de querer utilizar un modelo clasificador esto puede significar un problema ya que son 201 clases a predecir si se quiere aprender con la mayor granularidad posible, lo cual supone que se debe tener una cantidad adecuada de datos de entrenamiento para cada una de esas clases, como se puede apreciar en la Figura 6.2. Además de tener una cantidad de elementos desbalanceada de cada clase, no se cuenta con la cantidad suficiente de datos para analizar el problema con tanto nivel de detalle y es por ello que se plantean tres variantes para la reestructuración del atributo objetivo: ***aproximación a la decena más cercana***, ***distribución según calidad*** y ***distribución según potencial***. En el primer apartado del Anexo D se encuentran detalladas algunas pruebas realizadas que motivaron estos cambios en la definición del atributo objetivo.

Figura 6.2: Distribución original de porcentajes de puntos en respuestas

Aproximación a la decena más cercana

En primera instancia, la opción más directa sin que se pierda demasiado nivel de detalle fue la de agrupar el valor objetivo a la decena más cercana. La Figura 6.3 refleja estos cambios, con los cuales se obtuvieron 21 clases para que los modelos aprendan. Esto logró reducir la ausencia de datos en las clases, pero aun así la cantidad de datos disponible no es la suficiente para valores menores a 0.

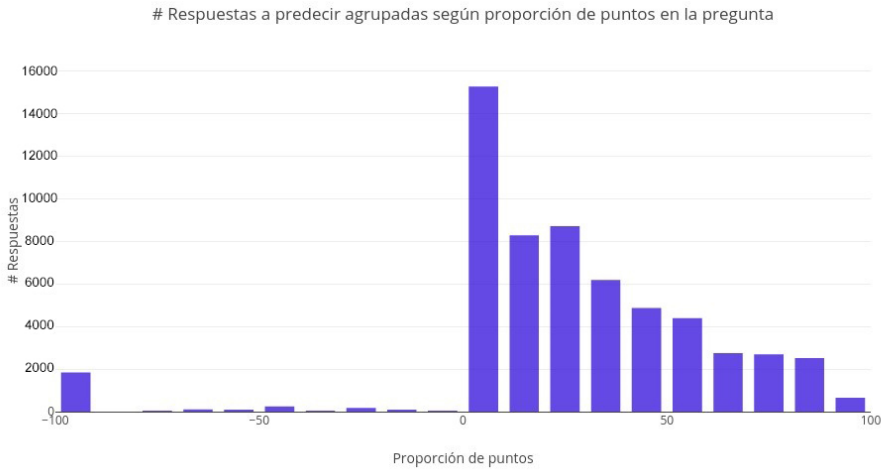

Figura 6.3: Distribución de respuestas según la decena más cercana

Distribución según calidad

Se definen distintos niveles de calidad de una respuesta a una pregunta —*muy mala*, *mala*, *no aporta*, *buena*, *muy buena* y *excelente*—. En la Figura 6.4 se puede ver la nueva distribución de los valores del atributo objetivo.

Figura 6.4: Distribución de respuestas según calidad

En el Anexo D se detallan pruebas realizadas con el modelo SVM de clasificación, en las que se pudo ver que no se tuvo un gran avance a la hora de aprender

las respuestas malas, dado que sigue siendo baja la cantidad de respuestas con estos puntajes. Por esta razón es que se define la última distribución:

Distribución según potencial

Con un menor nivel de granularidad se define una distribución donde se pretenden clasificar las respuestas *malas* y *sin aporte* por un lado, las *potencialmente buenas* y las que se consideran *buenas*, debido a que tienen una buena porción de puntos y ello ya bastaría para ser considerada una respuesta aceptable. Se toman como respuestas *malas* aquellas con puntaje negativo o 0; respuestas *potencialmente buenas* aquellas con menos del 22 % de los puntos repartidos entre las respuestas, y como *buenas* aquellas con un porcentaje mayor a este.

Figura 6.5: Distribución de respuestas según potencial

El valor seleccionado para dividir estas categorías se toma con el fin de obtener una distribución más balanceada en los datos disponibles, como se puede ver en la Figura 6.5.

Análisis de atributos

Luego de las pruebas detalladas en la sección anterior, se decide hacer un análisis de los atributos utilizados en el entrenamiento para determinar el impacto de cada uno en el desempeño del modelo. Se detallan a continuación las conclusiones obtenidas de este análisis. Los detalles de su desarrollo se pueden ver en la segunda parte del Anexo C.

Para todos los atributos se generan gráficas de barras cuyo eje horizontal contiene los valores del atributo mientras que en el eje vertical se representan los falsos positivos y verdaderos positivos con distintos colores. Para cada una de estas gráficas se pretende capturar la incidencia del atributo en el modelo entrenado, de acuerdo a las distribuciones presentadas. De este análisis, y como se puede apreciar en el segundo apartado del Anexo C, se desprenden dos conclusiones: la primera es que se debe cambiar el dominio de los atributos con los que se entrena el modelo, dado que son valores reales y continuos y presentan en muchas ocasiones valores aislados. Esto hace que el modelo no termine de capturar exactamente los valores adecuados de cada atributo, por lo que se piensa en categorizar el dominio de los atributos para facilitar esta tarea. La segunda conclusión lleva a la necesidad de la construcción de atributos nuevos que representen información de calidad de un par pregunta-respuesta, dado que individualmente se considera que estos atributos no aportan al modelo.

Modelos de aprendizaje automático

Entre las etapas de filtrado y preprocesamiento de los datos de Stack Exchange, de la extracción de sus atributos y posterior unificación y normalización, se fueron implementando distintas pruebas con un clasificador SVM, como se puede ver en el Anexo D. Una vez desarrollada la base de estas pruebas y finalizada esta etapa de extracción de datos, se desarrolla una clase con todas las herramientas y parámetros necesarios para entrenar distintos modelos de clasificación o regresión.

Esta clase toma como parámetros de entrada la ruta del archivo con los datos de entrada, la distribución del atributo objetivo que se quiere analizar y el modelo que se utilizará. Además, opcionalmente se puede indicar el archivo de configuración del cual extraer los atributos o los parámetros con los cuales entrenar el modelo seleccionado; se puede optar por utilizar pesos por ejemplo o por clase y persistir o cargar datos ya entrenados. Luego, se pueden obtener estadísticas sobre la distribución de las respuestas a predecir, o su total; se puede imprimir la matriz de confusión, *accuracy*, *precision* y *recall*, en caso de trabajar con un modelo clasificador, o mostrar el error cuadrático medio para los casos de regresión. Por último, en caso de querer persistir los datos, cada modelo, resultado o conjunto de datos es guardado a disco y comprimido para ahorrar espacio.

Una vez se instancia la clase, el flujo de ejecución es el siguiente:

1. Se carga el archivo con los datos.
2. Se filtran de acuerdo al archivo de configuración elegido.
3. Se parten los datos en conjuntos de entrenamiento y de evaluación.
4. Se entrena el modelo elegido.
5. Se obtienen y analizan resultados tras instanciar el modelo entrenado con los datos de prueba.

A continuación se describen brevemente los modelos utilizados para implementar la solución.

Support vector machines

Como primera herramienta para trabajar sobre los atributos obtenidos se utiliza el modelo SVC de Scikit-learn,[5] el cual provee un modelo de clasificación basado en el modelo SVM. Sobre este modelo se prueban distintas alternativas de implementación y configuración, como las variantes de los atributos objetivos y los pesos por ejemplo o por clase, que son conceptos que se detallan en la Sección 6.2. Además, se utiliza también la clase SVM Regressor de Scikit-learn, que permite entrenar un modelo de regresión SVM para obtener valores reales en lugar de categorías, como se describe en la Sección 2.3.

Scikit-learn provee dos clases para entrenar modelos de clasificación y regresión: SVC y SVR.[6] Para el modelo SVM de clasificación —SVC—, la biblioteca ofrece los atributos descritos en la Tabla F.1, mientras que para un modelo SVM de regresión —SVR— se tienen los mismos parámetros descritos en la Tabla F.1, con la adición del descrito en la Tabla F.2, ambas ubicadas en el Anexo F. Los parámetros de ambos modelos pueden ser configurados mediante un archivo de configuración, de modo de cambiar dinámicamente los valores y poder así entrenar sin modificar el código base.

Luego de separar los datos disponibles en una relación 90 % - 10 % para entrenamiento y evaluación respectivamente, se entrena el modelo con el primer conjunto y luego se lo prueba con el segundo. Los resultados del modelo SVC se detallan en una matriz de confusión consistente de los valores reales y los valores obtenidos por el clasificador para cada clase definida por el atributo objetivo (Sección 6.2), los puntajes de *accuracy*, *precision* y *recall*. En esta matriz se pueden ver reflejadas las instancias que el clasificador predijo correctamente contra aquellas que clasificó erróneamente. Para los resultados obtenidos por el modelo SVR se utiliza el error cuadrático mínimo, con el cual se minimiza el error del modelo.

Gradient boosting

Como se explica en la Sección 2.3, el modelo *gradient boosting* se construye a partir de un conjunto de *árboles de decisión* encadenados. Tras entrenar por primera vez al modelo, en cada etapa posterior se generan nuevos árboles que procuran enfocarse en los puntos débiles del árbol anterior. De esta manera se suele ajustar con mayor precisión al conjunto de entrenamiento en el correr de las iteraciones, con una posibilidad latente de sobreajuste a los datos de entrenamiento. Se utilizan las clases Gradient Boosting Classifier[7] y Gradient Boosting Regressor[8] provistas por

[5] http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[6] http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html
[7] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.htmll
[8] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.htmll

Scikit-learn, en las que de manera análoga al modelo SVM se clasifican las instancias disponibles.

En la Tabla F.3 ubicada en el Anexo F, se detallan los parámetros disponibles para los modelos Gradient Boosting Classifier y Gradient Boosting Regressor. Como para SVM, estos parámetros son configurables mediante archivos independientes al código implementado.

Un detalle importante de este modelo es que Scikit-learn provee un método para calcular las predicciones en las distintas etapas de entrenamiento, de modo que sea posible estudiar el comportamiento de la función de pérdida a medida que se agregan más estimadores, y también llevar un control de qué tanto se sobreajusta el modelo a los datos de entrenamiento, en comparación con los de prueba.

Redes neuronales

La biblioteca Scikit-learn provee modelos de redes neuronales —MLPClassifier[9] y MLPRegressor[10]—, mediante los cuales, dada una definición de parámetros como los descritos en la Tabla F.4, ubicada en el Anexo F (ambos modelos comparten el mismo conjunto de parámetros seleccionables), y un conjunto de instancias de entrenamiento, se calculan los pesos que componen una red neuronal. Cabe mencionar que, para muchas pruebas durante la integración de este modelo, los resultados obtenidos no lograban contemplar el aprendizaje de más de una clase objetivo; luego de varias pruebas con combinaciones de parámetros se logró obtener un modelo que medianamente comenzara a aprender y ajustarse a las instancias provistas. Los parámetros claves en esta definición fueron: la función de activación (*tanh*), la función de optimización (*sgd*) y la tasa de aprendizaje (*adaptive*).

Análisis de resultados y correcciones

Una vez integrados los modelos de Scikit-learn al flujo de la solución, se realizan diversas pruebas en búsqueda de lograr una mejora en los resultados. Entre ellas se encuentra la utilización de pesos, tanto por ejemplo como por clase, para un modelo SVM, para ver si esto causa sobreajuste en los modelos; se estudia el desempeño de los modelos con una cantidad variable de datos de entrenamiento y se investiga la curva de Precision-Recall en búsqueda de analizar el comportamiento de los modelos y así obtener una mejora en sus parámetros. A continuación se brinda una breve descripción de las pruebas, con mayor nivel de detalles en el Anexo F.

Pruebas con pesos por ejemplo y pesos por clase

El parámetro de pesos por clases para el modelo SVM permite asignar determinada relevancia a los elementos de una clase; en particular se utiliza cuando se

[9] http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

[10] http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html

cuenta con un conjunto de datos desbalanceados, es decir, se tienen más atributos de una o algunas clases que de otras. Esto se realiza asignando un peso inversamente proporcional a la frecuencia de una clase en el conjunto de datos, lo que le da mayor relevancia a las clases que tienen menor frecuencia. En el primer apartado del Anexo E se brinda una descripción con mayor profundidad de lo implementado.

El parámetro de pesos por ejemplo permite asignar determinada relevancia a un ejemplo del conjunto de datos. Se utiliza con el objetivo de resaltar un ejemplo que cumpla con determinadas características que se quiere que el clasificador aprenda; por ejemplo, puede ser de interés enfocar la atención en instancias que presentan una gran cantidad de entidades nombradas o presentan una distancia de Hellinger cercana a 0. En esta solución se implementa una función que en base a determinadas características se calcula un peso para cada ejemplo, pero luego de diversas pruebas se corroboró que la utilización de dicha función genera un sobreajuste de los datos de entrenamiento, por lo que se terminó por descartar. En la primera parte del Anexo E se pueden encontrar más detalles al respecto.

Relación entre cantidad de datos y precisión del clasificador

Dada la cantidad total de instancias disponibles de entrenamiento, interesa saber si al aumentar esta cantidad —ya sea mediante la selección de otro subforo con mayor volumen de datos o la utilización de datos actuales—[11] se da una mejora tangible en el desempeño de la solución. Para esto, se realizan pruebas con una cantidad variable de datos de entrenamiento, partiendo del 10 % y aumentando gradualmente hasta llegar al 100 %.

Pruebas realizadas sobre un clasificador SVM y el atributo objetivo convertido según la distribución de potencial indican que se da una ínfima mejora al agregar datos de entrenamiento, por lo que no se centra el esfuerzo en conseguir una mayor cantidad de datos a la hora de ejecutar las pruebas finales de la solución. En el segundo apartado del Anexo E se pueden encontrar las pruebas realizadas, detallándose el procedimiento y la conclusión a la que se arribó.

[11] Se recuerda al lector que los datos utilizados para el desarrollo de esta solución datan de setiembre de 2014, como se indica en la Sección 4.1.

Capítulo 7

Ejecución y resultados

Para lograr un conjunto de pruebas y resultados consistentes, se plantea la ejecución de un conjunto de casos particulares para los modelos definidos en el Capítulo 6.

Dado que se cuenta con una elevada cantidad de atributos para entrenar los modelos de aprendizaje automático desarrollados, se definen distintos subconjuntos de atributos con los cuales se pretende maximizar los resultados e interpretar la causa de dicha mejoría. Para cada caso se contrastan las pruebas con distintos dominios de información —valores normalizados y valores categorizados mediante la técnica *one hot encoding*—, y además se mide el desempeño de la solución luego de redefinir los valores objetivo —de mayor a menor granularidad en las clases a predecir—.

A la hora de entrenar los modelos, del total de instancias disponibles se toma el 90 % para entrenar y el 10 % restante se divide a la mitad, una para validar lo aprendido por cada modelo, en tanto que la segunda mitad se utiliza para evaluar las instancias luego de la búsqueda de hiperparámetros para los modelos con los que se tuvieron los mejores resultados.

A continuación se brinda una descripción del módulo utilizado para gestionar los atributos utilizados, se detallan las pruebas realizadas y se exponen los mejores resultados obtenidos en cada una.

7.1. Configuración

El objetivo principal del módulo de configuración es permitir la selección de distintos atributos, e incluso distintas opciones de procesamiento, desde un lugar centralizado y sin necesidad de modificar el código fuente. Se genera un archivo de configuración, el cual especifica para cada atributo si está habilitado o no; luego y previo a cada entrenamiento se consulta este archivo para determinar cuáles atributos son tomados en cuenta. Esta decisión se toma para poder variar de forma eficiente la combinación de atributos con la que se entrena. Esto permite estudiar el comportamiento al entrenar bajo un determinado conjunto de atributos o potencialmente encontrar la combinación de atributos que optimicen el desempeño del entrenamiento. Dentro del archivo de configuración se pueden encontrar los atributos agrupados por tipo, cada tipo está precedido por un numeral y seguido por todos los atributos que pertenecen a él, como se puede ver en el Ejemplo 7.1.

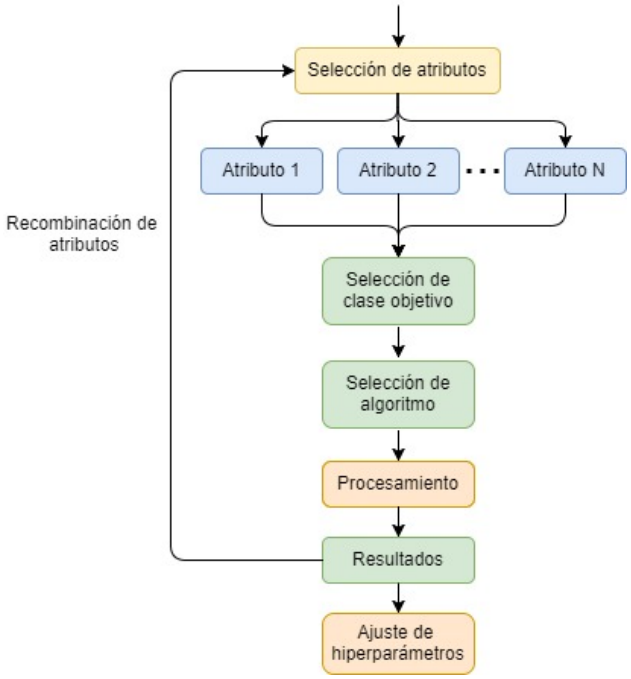

Figura 7.1: Ejecución de la solución

```
# Cosine similarity
q_ans_cos_sim = True

# Metadata features
ans_user_bronze_amount = True
...
q_user_silver_amount = True

# XML Features
q_href_marks = True
...
ans_exclamation_marks = True

# Topic models
hellinger = True
jaccard = True
kullback_leibler = True

# Syntactic similarity
q_verb_count = True
...
ans_tags = False

# Direct attributes
q_creation_date_days = True
...
```

```
ans_user_down_votes_count = True

# Syntactic attributes
q_sentences_count = True
...
ans_sentence_average_length = True


[SVM_Parameters]
svm_c = 1.0
...
svm_verbose = False
```

Ejemplo 7.1: Archivo de configuración

7.2. Definición de pruebas y medidas

A la hora de ejecutar las pruebas, se tienen cinco variables para tomar en cuenta: *conjunto de atributos*, *dominio de atributos*, *clases de atributo objetivo*, *modelo de aprendizaje automático* y *tipo de problema*.

Para la selección de los atributos a utilizar por los modelos, como se detalla en el Anexo B, a gran escala las agrupaciones de atributos disponibles son: *atributos de cantidad, longitud, distancia, reputación de usuarios, fechas, calidad y cantidad entre pares de preguntas y respuestas*.

Se definen **tres combinaciones** que contemplan la selección de distintos conjuntos de atributos. Para la **primera** se optó por hacer hincapié en los atributos de cantidad y calidad entre respuestas y preguntas frente a los atributos independientes, como cantidad de palabras en negrita, cursiva, cantidad de enlaces, etc. El resto de los atributos se utilizan por completo. Para la **segunda combinación** se prueba lo opuesto a la primera respecto a los atributos de calidad y cantidad, donde se hace hincapié en los atributos originales. Para la **tercera** y última, se utilizan los mismos atributos que en la primera combinación con la diferencia de que se deshabilitan aquellos atributos que guarden alguna relación con los usuarios que crearon la pregunta o sus respuestas. Las dos primeras combinaciones buscan reflejar la importancia de encontrar una relación semántica al evaluar la calidad de una respuesta, y por otro lado intentan reflejar conexiones entre atributos de pregunta y sus respuestas, como se concluyó tras el análisis reflejado en la Sección 6.2. Para la tercera prueba lo que se busca reflejar es la importancia de los atributos de usuario para estudiar el comportamiento de la solución frente a datos que no posean información de usuario.

Una vez definidas las combinaciones y sus *conjuntos de atributos*, se deben definir sus *dominios de atributos*. Se seleccionan dos tipos para cada una: **atributos normalizados** a partir de sus valores originales, según se detalla en la Sección 5.4, y atributos convertidos mediante la técnica **OHE**, donde los valores de los atributos originales se agrupan en clases, y luego se genera un atributo booleano independiente

por cada una de estas clases. En el Anexo B se pueden ver estos valores.

El tercer paso para definir las pruebas consiste en seleccionar las *clases del atributo objetivo*. Como se detalla en la Sección 6.2, estas consisten en: **distribución según decena más cercana al porcentaje de puntos en una respuesta**, **la calidad de la respuesta** y **el potencial de una respuesta**.

Finalmente, se seleccionan los *modelos de aprendizaje automático* y los *tipos de problemas* a aprender. Como se describe en la Sección 6.2, los modelos que se ejecutarán son: **Support Vector Machines**, **Gradient Boosting** y **Multi-layer Perceptron**. Cada uno de estos modelos se desarrolla en sus variantes de **clasificación** y de **regresión**; con el primer tipo de problema se busca encasillar las respuestas en clases dadas, mientras que con regresión se busca brindar objetivamente un porcentaje de puntos a una respuesta, y luego compararla con los valores originales.

A modo de resumen, se tienen un total de **combinaciones de atributos**, cada una con **dos tipos de dominio de atributos**, **tres distribuciones del atributo objetivo**, **tres modelos** y **dos tipos de problemas** para cada uno. En total se tienen **108 instancias de ejecución** individuales para comparar los modelos seleccionados, las cuales se pueden ver en el Anexo F, donde están agrupadas según modelo y número de combinación.

Medidas de evaluación

A la hora de evaluar los resultados obtenidos, para los modelos de clasificación se decide utilizar la medida ***accuracy***, mientras que para los modelos de regresión se utiliza la medida ***mean squared error*** **(MSE)** y puntaje *R2*. El significado de estas medidas y cómo interpretarlas está fuertemente ligado al dominio de la clase objetivo, la definición de la medida y la combinación realizada. Si bien medidas como ***precision*** o ***recall*** son muy importantes, a la hora de evaluar primariamente los resultados se utiliza ***accuracy***.

Ranking

Al buscar una medida estandarizada que permita comparar los modelos de clasificación con los de regresión no se encontró una que permita representar los resultados obtenidos de forma intuitiva al lector. Es por esta razón que se definió una medida llamada *ranking*, cuyo objetivo es comparar objetivamente estos resultados obtenidos de la ejecución de los modelos planteados en la Sección 2.3. Esta medida se define como el *porcentaje de acierto de la comparación del orden de la lista original de respuestas con la predicha por un modelo de aprendizaje automático*.

Antes de profundizar en el desarrollo de la medida de *ranking*, es necesario mencionar que al momento de investigar estas herramientas se priorizó encontrar una medida que permita interpretar los resultados obtenidos de manera intuitiva, preferiblemente un porcentaje que refleje la cantidad de respuestas predichas correctamente

comparada con los valores originales. En primera instancia se consideraron las medidas *cumulative gain* y *discounted cumulative gain*, usualmente utilizadas a la hora de evaluar listados con un orden específico de elementos, pero erróneamente se consideró que no se ajustaba a las necesidades planteadas dados los resultados esperados por estas y se las descartó. Los valores obtenidos dependen, entre otras cosas, de la cantidad de respuestas en una pregunta o de un peso específico definido para una determinada calidad de respuesta, por lo que se esperó un valor arbitrario difícil de interpretar y se decidió implementar una medida simple que lograra el cometido buscado. Sobre el cierre de este proyecto se encontró la medida *normalized discounted cumulative gain* [50] que define un valor óptimo con el cual comparar los valores obtenidos y así obtener un porcentaje de referencia. Dado el corto plazo disponible entre esta observación y la defensa del trabajo, como se menciona en el Capítulo 8, se deja como trabajo a futuro estandarizar con esta medida los resultados obtenidos por los modelos entrenados.

Como se describe en la Sección 5.3, la posición de una respuesta en la lista de respuestas a una pregunta está estrechamente relacionada con el porcentaje de puntos que esta recibe, dado el total de puntos repartidos entre las respuestas. Se define como *mejor respuesta* a la que se encuentre en el primer lugar de la lista de respuestas, asumiendo un orden decreciente respecto al porcentaje de puntos recibidos.

Para establecer un criterio de comparación del orden de los valores de las respuestas predichas con la lista original, se asume este orden decreciente, se compara cada par de elementos predichos de la lista con sus equivalentes originales y se contabiliza la cantidad de aciertos sobre la cantidad de intentos. Este valor se denomina *tasa de acierto* de la lista de puntajes predichos.

Se denota la importancia del *orden entre pares de elementos*, ya que la comparación no solo se realiza entre los elementos de las listas, sino que además se compara internamente cada par de elementos de la lista predicha con su correspondiente par en la lista original. Se destaca esta diferencia para contemplar casos de interés en el análisis de los resultados, que con una simple comparación posicional entre dos listas no se pueden ver reflejados, como se observa en el ejemplo a continuación.

Se definen dos listas, cada una representando la lista original y la predicha, de los valores de porcentaje de puntos totales obtenidos para cada respuesta en una pregunta. Para la lista original se tiene que la primera respuesta posee un 50 % de los puntos repartidos entre todas, seguida por una respuesta con el 30 % de los puntos, y finalmente la última respuesta con el 20 %. La lista de respuestas predichas presenta, para estas mismas respuestas, los valores de 15 %, 50 % y 35 % respectivamente. En la Tabla 7.1 se pueden ver reflejados estos valores.

Si se realiza una comparación posición a posición se puede ver que la tasa de acierto es nula, ya que no se presentan igualdades entre los valores originales y los predichos por el sistema. Sin embargo, se puede ver que se cumplen algunas relaciones, como por ejemplo que la tercera respuesta presenta una calidad menor que la segunda, característica que se ve reflejada en ambas listas y que es considerada al utilizar la medida de *ranking*.

Respuesta	Puntos originales	Puntos predichos
Respuesta 1	50 %	15 %
Respuesta 2	30 %	50 %
Respuesta 3	20 %	35 %

Tabla 7.1: Ejemplo de medida de *ranking*

Para calcular esta tasa de acierto, se toma cada valor de la lista predicha y se lo compara con el elemento siguiente de la lista. Luego, se compara la relación entre ambos con el par de valores de la lista original, y se continúan evaluando los pares de elementos restantes. Se toma la suma de cada acierto y se lo divide entre la cantidad total de combinaciones, obteniéndose así un porcentaje final que compara las listas de respuestas a una pregunta. Finalmente, se toman todas las preguntas a analizar, se aplica la medida y se suman los porcentajes obtenidos, tras lo cual se promedia con la cantidad de preguntas analizadas.

Este ejemplo en particular presenta un valor de *ranking* de un 33 %, contra un 0 % que se obtendría en caso de comparar posicionalmente los valores de las listas. Además, de esta forma se captura la relación entre los valores y no necesariamente se esperan valores idénticos. Cabe mencionar que además de la primera aproximación mediante una comparación posicional de las listas, se consideró utilizar medidas como la *distancia mínima de edición* —también conocida como *distancia de Levenshtein*—, frecuentemente utilizada para saber qué tan similares son dos listas de objetos, y otras variantes que no terminaron de reflejar lo que se quiso evaluar, por lo que se procedió a utilizar lo definido como medida definitiva para esta solución.

A continuación se realiza un análisis de los resultados obtenidos tomando en cuenta las variables definidas en esta sección, con el fin de determinar qué modelo presentó un mejor desempeño, de acuerdo al tipo de problema, atributos utilizados y clase objetivo. Para cada modelo se presentan dos tablas —una para cada dominio de atributo— junto con una breve justificación. Luego, un resumen general para cada modelo da lugar a un análisis general en la Sección 7.8. Por último, para los mejores resultados se realiza una búsqueda de hiperparámetros con el fin de optimizar los atributos del modelo.

Nota: las figuras expuestas para cada modelo representan las matrices de confusión obtenidas para las tres combinaciones definidas en la Sección 7.2, para cada dominio del atributo objetivo. Dado el tamaño de estas, para lograr una correcta apreciación es necesario que se visualicen a escala real, razón por la cual se limita la cantidad de figuras expuestas para facilitar la lectura del trabajo. En el Anexo F se pueden encontrar todas las figuras pertenecientes a este capítulo.

7.3. SVM

A continuación se describen las ejecuciones de los modelos de clasificación y regresión de *support vector machines* utilizados para predecir el porcentaje de los

puntos totales de cada respuesta perteneciente a una pregunta, así como su orden relativo respecto a este atributo.

Clasificación

Los modelos SVM de clasificación son entrenados con los parámetros especificados en la Tabla F.5, ubicada en el Anexo F.

Los resultados obtenidos tras las ejecuciones se pueden visualizar en las Tablas 7.2 y 7.3, en las que para cada clase objetivo se obtienen los puntajes de *accuracy* y de *ranking* en cada combinación.

	Combinación 1		**Combinación 2**		**Combinación 3**	
Clase objetivo	**Acc.**	**Rank.**	**Acc.**	**Rank.**	**Acc.**	**Rank.**
Porcentajes	19.84 %	35.0 %	18.59 %	**36.0** %	19.59 %	33.0 %
Calidad	26.05 %	**41.0** %	25.27 %	38.0 %	23.27 %	37.0 %
Potencial	43.30 %	49.0 %	45.17 %	48.0 %	43.86 %	**50.0** %

Tabla 7.2: Resultados por clase para SVM de clasificación con atributos normalizados

	Combinación 1		**Combinación 2**		**Combinación 3**	
Clase objetivo	**Acc.**	**Rank.**	**Acc.**	**Rank.**	**Acc.**	**Rank.**
Porcentajes	23.46 %	**59.0** %	22.68 %	56.9 %	22.21 %	59.0 %
Calidad	37.34 %	**57.9** %	36.85 %	55.0 %	36.72 %	56.9 %
Potencial	60.90 %	**59.0** %	60.96 %	56.9 %	60.34 %	57.9 %

Tabla 7.3: Resultados por clase para SVM de clasificación con *one hot encoding*

Para la distribución según decena más cercana del porcentaje de puntos de una respuesta —de aquí en más se la menciona como distribución según **porcentajes** a modo de simplificación—, en la primera fila de las Tablas 7.2 y 7.3 se encuentran los valores obtenidos para la distribución según porcentajes, definida en la Sección 6.2. Esta distribución de atributo objetivo presenta la mayor granularidad posible dentro de lo definido, ya que procura agrupar el porcentaje de puntos esperado para una respuesta a la decena más cercana. Esto pretende facilitarle el trabajo al modelo, dado que si no serían demasiadas clases posibles para las instancias disponibles (201 contra 21).

A nivel de clasificación individual se tiene que para la primera combinación se alcanza el máximo valor de *accuracy* (19.84 %) con los valores normalizados, y aumenta con los valores transformados a OHE (23.46 %). A nivel colectivo se mantiene esta tendencia: para valores de OHE se obtuvo un mejor valor de *ranking* (59.0 %) frente al 35.0 % reportado por el dominio normalizado, por lo que prácticamente lo duplica.

En primer lugar, la mejora a nivel individual puede deberse a la nueva definición de los datos tras aplicar OHE, ya que de esta manera se agrupan los valores de acuer-

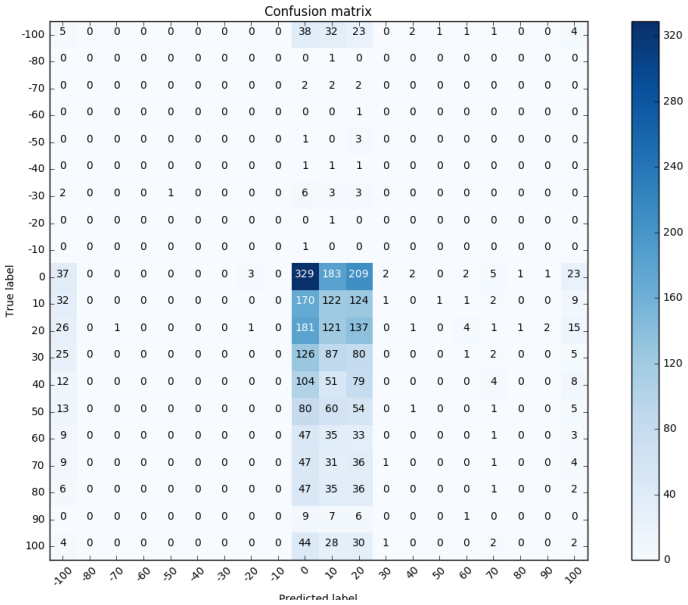

Figura 7.2: Modelo SVM, atributos normalizados y clases según porcentajes. Segunda combinación de datos

do a una lógica determinada y no se deja que el modelo los interprete libremente. Por más que en primera instancia los datos se encuentran normalizados, y con esto se evitan saltos muy pronunciados al analizarlos, en muchos casos se encuentran valores aislados y muy lejanos a la media que pueden resultar molestos e introducir ruido al clasificador. Luego, la mejora a nivel colectivo para las combinaciones en la Tabla 7.3 son consecuencia directa del aumento de *accuracy* de los clasificadores: cada clasificador en cada ejecución mostró mejor desempeño que su contraparte de la Tabla 7.2.

Para los atributos normalizados se puede apreciar que en la matriz de confusión de la segunda combinación —visualizada en la Figura 7.2— se tiene una agrupación de instancias predichas cercana a la zona media, donde las respuestas o no aportan o tienen pocos puntos. Para las combinaciones primera y segunda, se puede ver —en el Anexo F.3— que no existe esta tendencia de agrupación hacia el centro, sino que está apenas más distribuida hacia la derecha. Dado que en una matriz de confusión se evalúa qué tan confuso se encuentra el modelo luego de la evaluación con los datos de prueba, al cruzar los valores predichos con los reales es esperable una diagonal desde el borde superior izquierdo hacia el inferior derecho, en la cual las instancias se agrupen uniformemente. Una mayor cantidad de valores agrupados en columnas denota un modelo que no sabe distinguir entre los valores verdaderos y clasifica la mayor parte de las instancias como un único valor. Una agrupación a nivel de filas demuestra que el modelo tampoco tiene un criterio claro para un solo valor verdadero, ya que para el mismo valor objetivo clasifica con valores diferentes.

Para los atributos transformados a OHE se nota un comportamiento mucho más parejo, como se puede apreciar en la Figura 7.3. Esto es positivo a la hora evaluar datos sin atributos de usuario ya que presenta valores muy similares, quizás no tan certeros individualmente, pero sí a nivel colectivo. Otra diferencia encontrada respecto a los resultados para atributos normalizados es que se intentó predecir con

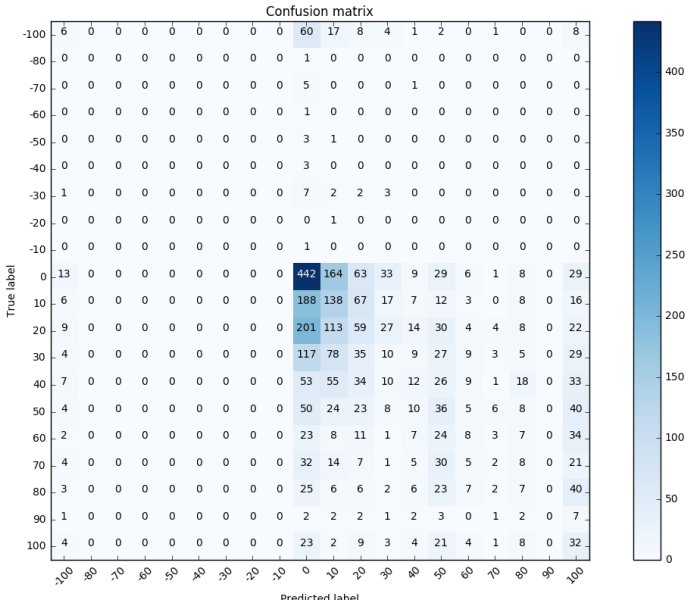

Figura 7.3: Modelo SVM, atributos OHE y clases según porcentajes. Primera combinación de datos

mayor frecuencia las respuestas excelentes —valores cercanos al 100 % —, lo cual denota cierta tendencia a aprender características de buenas respuestas. No se ve este comportamiento para las respuestas malas —con puntaje negativo—, lo cual puede deberse a la escasa cantidad de datos de este tipo.

Para la segunda distribución utilizada —distribución según **calidad**—, como se describe en la Sección 6.2, se opta por cambiar los valores numéricos de porcentaje de puntos predichos por un concepto asociado a la interpretación de dichos porcentajes, y de acuerdo a este concepto se agrupan los valores y se los categoriza de acuerdo a la calidad de las respuestas.

En general se nota a nivel individual un aumento en el porcentaje de acierto para ambos dominios de atributos, mientras que a nivel colectivo se puede apreciar que decrece el mejor resultado para atributos con OHE respecto a la distribución según porcentajes, pero crece para la clasificación con los datos normalizados. El aumento individual es esperado dada la reducción de las clases objetivo: el modelo no necesita ser tan preciso para encontrar relaciones, lo cual hace que aumenten las probabilidades de acertar en la predicción. Este comportamiento resta detalle a la hora de estudiar las respuestas, ya que en lugar de obtener un valor objetivo se tiene uno subjetivo, que es la calidad de una respuesta. Cabe mencionar que, si bien tanto a nivel individual como colectivo la tercera combinación obtuvo peores resultados que la primera, al analizar la matriz de confusión se puede ver una mejor distribución de las instancias predichas alrededor de los valores de respuesta excelentes. Esto se puede interpretar en el sentido de que para valores excelentes puede no existir una dependencia fuerte de los atributos de usuario.

A nivel colectivo se observa la disminución de la tasa de acierto medida en *ranking* para los valores de OHE comparados con la distribución según porcentaje. La

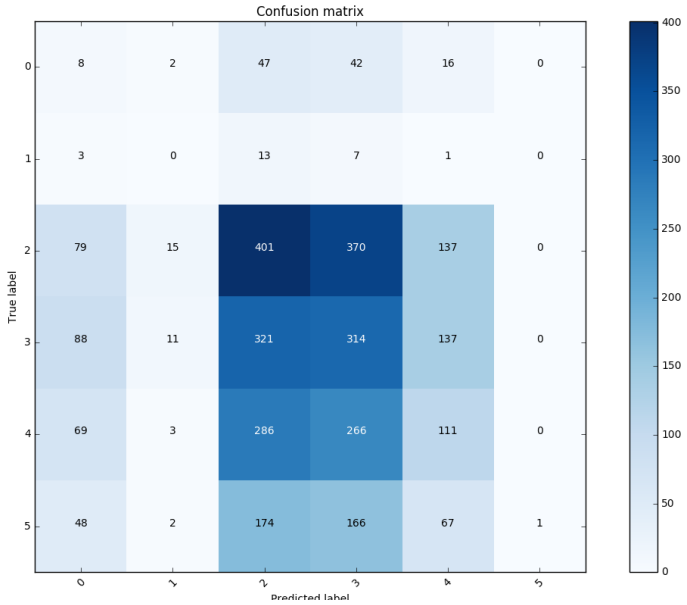

Figura 7.4: Modelo SVM, atributos normalizados y clases según calidad. Primera combinación de datos

razón de este comportamiento puede deberse a que la ganancia obtenida tras convertir los datos a OHE se ve deteriorada al disminuir la cantidad de clases objetivo, ya que existe una mayor probabilidad de que se superpongan valores similares y se dé un orden incorrecto de las respuestas. Este comportamiento no se aprecia en la distribución anterior ya que se tiene más holgura a la hora de comparar el orden de dos respuestas debido a la cantidad de clases presentes.

Para los atributos normalizados, para las tres combinaciones se nota cómo el modelo captura características de respuestas muy malas de manera incorrecta, ya que intenta clasificar como tales a un buen número de respuestas buenas. Si bien esto se presenta en menor cantidad tras utilizar los atributos categoriales, no se presenta en ningún momento un aumento en la predicción de respuestas malas o muy malas, por lo cual para ambos dominios de atributos se puede afirmar que el modelo no reconoce este tipo de respuestas. Un ejemplo de esto se puede apreciar en la Figura 7.4.

Para el dominio con OHE se puede notar una mayor relevancia de las respuestas excelentes, lo cual sugiere que al categorizar los atributos se pueden interpretar de mejor manera los atributos relevantes, mientras que al trabajar sobre sus valores originales —aunque estén normalizados—, la diversidad de valores probablemente confunda al modelo, permitiéndole determinar con mayor seguridad las respuestas muy malas.

Por último, para la distribución según **potencial** se puede observar que al reducir aún más la diversidad de clases es lógico que se dispare el nivel de acierto, ya que el modelo tiene mayor probabilidad de aprender características relevantes de las respuestas, pero no determinantes como para discriminar con mejor detalle. Con esta distribución se llega a casi un 45.17 % de acierto individual y 50.0 % como máximo de valor de *ranking* para datos normalizados, mientras que para los datos

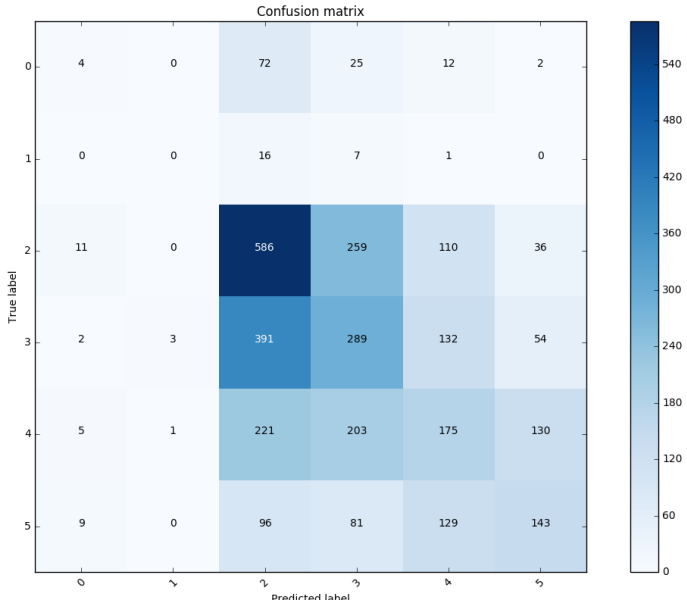

Figura 7.5: Modelo SVM, atributos OHE y clases según calidad. Primera combinación de datos

categoriales se llega a un máximo de 60.96 % de *accuracy*. El comportamiento apreciado en las Figuras 7.6 y 7.7 va de la mano con lo visto para las otras distribuciones, con la misma tendencia respecto a las agrupaciones de instancias predichas.

Para los atributos normalizados, la Figura 7.6 muestra cómo se agrupan las instancias en el cuadrante inferior derecho casi en la misma proporción para las clases correctas como incorrectas. Esto no aporta en el análisis de este modelo, ya que aunque haya mejorado respecto a la distribución anterior, no se puede ver que el modelo haya aprendido alguna clase con propiedad. Distinto es el caso de los atributos categoriales, con los que si bien no se llega a aprender características de preguntas malas, se puede ver que para buenas y potencialmente buenas se logra marcar diferencia.

En líneas generales, para los valores de OHE se presentan dos tendencias: la primera es que superan con creces los valores obtenidos en comparación con los atributos normalizados, destacándose la primera combinación frente al resto, mientras que la segunda es que para las tres combinaciones en la Tabla 7.3 se obtuvieron valores muy similares a nivel colectivo, a diferencia de lo encontrado en la Tabla 7.2. Una posibilidad es que esto ocurra porque al establecer clases y agrupar los valores de los atributos con dominio OHE esto permite uniformizar los datos, haciendo que valores que se consideraban aislados pertenezcan a una misma clase, lo que fue uno de los objetivos a los que se apuntó al calcular los atributos de calidad y cantidad entre respuestas y preguntas.

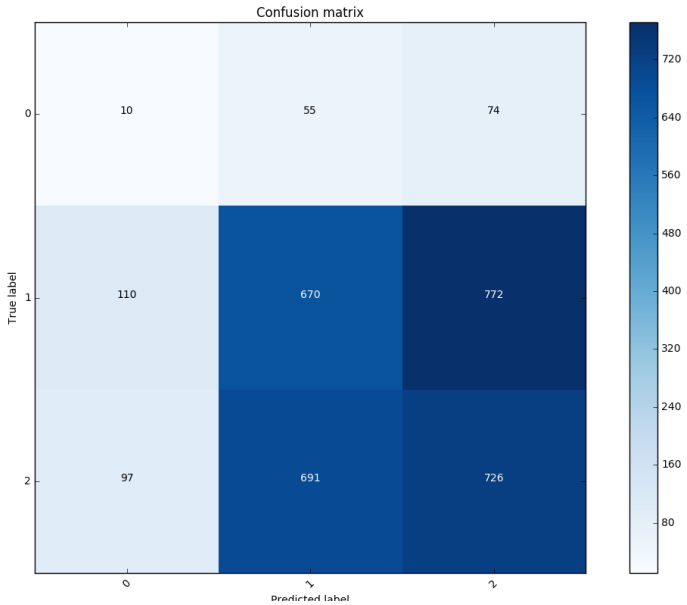

Figura 7.6: Modelo SVM, atributos normalizados y clases según potencial. Tercera combinación de datos

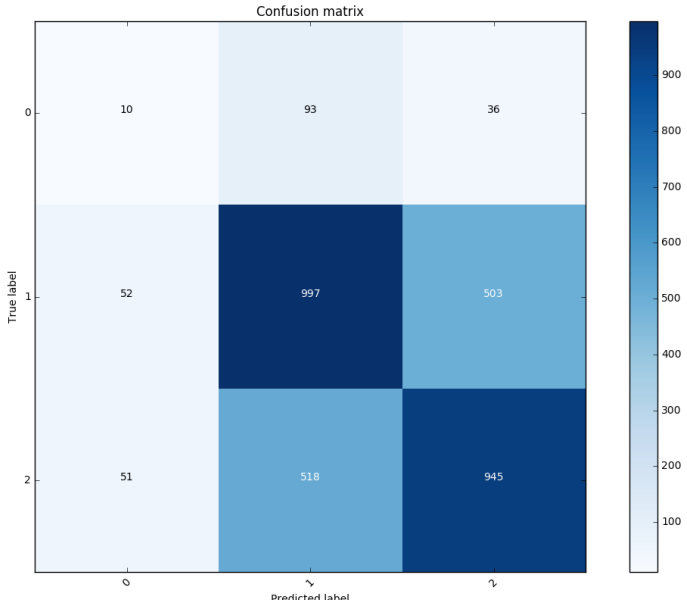

Figura 7.7: Modelo SVM, atributos OHE y clases según potencial. Primera combinación de datos

Regresión

Los modelos SVM de regresión son entrenados con los parámetros especificados en la Tabla F.6, ubicada en el Anexo F.

	Combinación 1			Combinación 2			Combinación 3		
Distrib.	**MSE**	**R2**	**Rank.**	**MSE**	**R2**	**Rank.**	**MSE**	**R2**	**Rank.**
Porcentajes	1303.69	-0.006	48.0 %	1303.67	-0.006	48.0 %	1303.61	-0.006	**50.0** %
Calidad	1.47	-0.011	**50.0** %	1.48	-0.014	50.0 %	1.47	-0.012	50.0 %
Potencial	0.38	-0.164	**49.0** %	0.39	-0.168	47.0 %	0.38	-0.158	48.0 %

Tabla 7.4: Resultados por clase para SVM de regresión con atributos normalizados

	Combinación 1			Combinación 2			Combinación 3		
Distribución	**MSE**	**R2**	**Rank.**	**MSE**	**R2**	**Rank.**	**MSE**	**R2**	**Rank.**
Porcentajes	899.62	0.305	**75.0** %	922.13	0.288	75.0 %	941.10	0.273	74.0 %
Calidad	0.94	0.355	74.0 %	0.93	0.361	**74.0** %	0.96	0.339	73.0 %
Potencial	0.25	0.225	67.0 %	0.25	0.237	**67.0** %	0.26	0.204	66.0 %

Tabla 7.5: Resultados por clase para SVM de regresión con *one hot encoding*

Como se observa en las Tablas 7.4 y 7.5 para la distribución según **porcentaje**, se obtiene un valor de MSE de 1300 con atributos normalizados y de 900 con OHE, mientras que el porcentaje de *ranking* está entre el 50 % y el 75 % con atributos normalizados y OHE respectivamente.

El dominio de la clase objetivo según porcentajes es [-100, -90, .. 0 .. , 90, 100] por lo tanto es normal obtener valores elevados de MSE ya que esta medida es directamente proporcional al tamaño del dominio de la distribución objetivo y representa el error promedio que se tiene al clasificar una determinada pregunta.

Aunque se cuente con un MSE elevado el **porcentaje** de *ranking* es bastante alto, y esto se debe a que si bien SVM no es capaz de predecir exactamente el valor, está prediciendo un valor cercano al valor real, y por lo tanto se está conservando la relación del porcentaje de puntos entre las respuestas a una determinada pregunta. De esto último se puede afirmar que si bien SVM no es la mejor opción para predecir el porcentaje de una pregunta individual, sí lo es a la hora de predecir el orden de las respuestas a su respectiva pregunta.

Luego, para la distribución según **calidad**, lo primero que se observa es que el MSE disminuye de forma sustancial respecto a la distribución anterior lo cual es coherente ya que la cantidad de clases bajó de 21 —según su porcentaje de puntos— a 6 —según su calidad— y el MSE es inversamente proporcional al tamaño del dominio. Tanto para los atributos normalizados como para los atributos con OHE el *ranking* se mantiene relativamente igual (pequeñas variaciones de 1 % o 2 %), por lo tanto se puede afirmar que la disminución del MSE se debe exclusivamente a la disminución del dominio de la clase. Otra observación importante es que a mayor MSE, menor porcentaje de *ranking*, lo cual es coherente ya que MSE es una medida

de error y *ranking* es una medida de rendimiento.

Respecto a la distribución según **potencial**, se reafirma lo observado en la distribución anterior: se tiene una disminución del MSE debido al nuevo tamaño del dominio —tres clases—, sin embargo existe una disminución importante del *ranking* con respecto a las clases anteriores (medianas variaciones de entre 7 % y 8 %). Si se tiene un dominio con tres atributos y un MSE de 0.4 esto quiere decir que el valor a estimar y el valor estimado se encuentran en promedio a una distancia de 0.4 en términos de MSE, lo que es positivo ya que resulta en una menor probabilidad de confundir la clase de una pregunta y el orden de las preguntas en el *ranking*.

Comparando las Tablas 7.4 y 7.5 se observa que los resultados son mejores y varían menos de combinación a combinación para OHE. Esto se debe a que OHE permite traducir los atributos de categoría sin añadirles una noción de orden, a diferencia de lo que pasa con la normalización, proceso en el que los atributos de categoría son traducidos a números que tienen una relación de orden entre ellos, que potencialmente —y es lo que pasa por lo general— podría no existir entre los atributos de categoría. La métrica R2, mencionada en la Sección 2.3, reafirma esta idea ya que se cuenta con valores positivos para las combinaciones con OHE, y valores negativos para las combinaciones con atributos normalizados; lo que señala que entrenar con OHE genera mejores modelos de regresión que entrenar con atributos normalizados.

7.4. Gradient Boosting

Se describen a continuación las ejecuciones de los modelos de clasificación y regresión de Gradient Boosting (GB) utilizados para predecir el porcentaje de los puntos totales de cada respuesta perteneciente a una pregunta, así como su orden relativo respecto a este atributo.

Clasificación

Los modelos GB de clasificación son entrenados con los parámetros especificados en la Tabla F.7, ubicada en el Anexo F.

	Combinación 1		**Combinación 2**		**Combinación 3**	
Clase objetivo	**Acc.**	**Rank.**	**Acc.**	**Rank.**	**Acc.**	**Rank.**
Porcentajes	21.37 %	31.0 %	21.93 %	**34.0** %	21.40 %	33.0 %
Calidad	26.21 %	37.0 %	27.51 %	38.0 %	27.70 %	**41.0** %
Potencial	48.95 %	48.0 %	49.01 %	47.0 %	48.51 %	**49.0** %

Tabla 7.6: Resultados por clase para GB de clasificación con atributos normalizados

	Combinación 1		Combinación 2		Combinación 3	
Clase objetivo	**Acc.**	**Rank.**	**Acc.**	**Rank.**	**Acc.**	**Rank.**
Porcentajes	24.62 %	**67.0** %	23.06 %	66.0 %	23.27 %	66.0 %
Calidad	38.84 %	**66.0** %	40.34 %	65.0 %	40.68 %	66.0 %
Potencial	66.11 %	62.0 %	67.36 %	**63.0** %	65.40 %	61.0 %

Tabla 7.7: Resultados por clase para GB de clasificación con OHE

Para la distribución según **porcentajes**, respecto al modelo anterior (SVM) se tiene una diferencia mayor tanto en el valor de *accuracy* de los datos normalizados como en el de los datos categoriales, siendo el segundo el más favorecido como se puede comprobar al comparar las Tablas 7.6 y 7.7. Las matrices de confusión no presentan demasiados cambios respecto a SVM por lo que se omiten en este apartado. Estas matrices, así como también el resto de las pertenecientes a este modelo, se pueden visualizar en el cuarto apartado del Anexo F.

Al comparar las ejecuciones, se encuentran similares las tres combinaciones para los atributos normalizados con respecto a la segunda combinación de la misma categoría y dominio en el modelo SVM. Se nota un leve aumento en el *accuracy* —cercano a un 2 %—, siendo la segunda combinación la mejor respecto al modelo anterior. A nivel colectivo los resultados son similares también, aunque levemente menores.

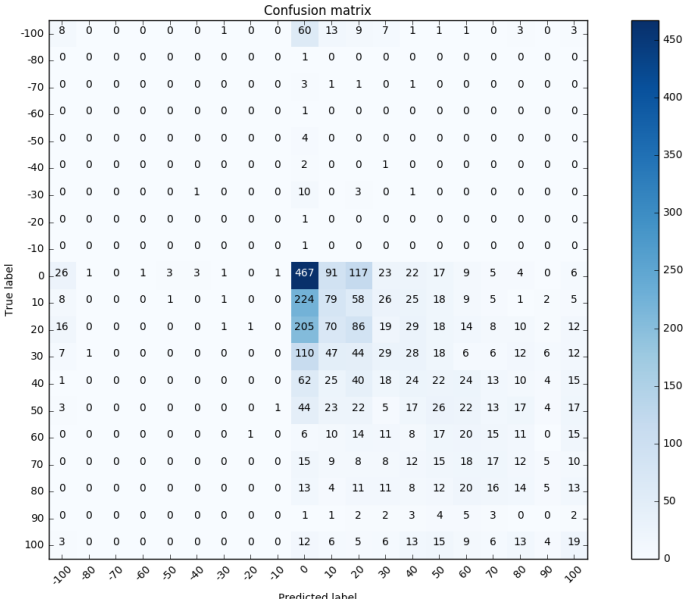

Figura 7.8: Modelo GB, atributos OHE y clases según porcentajes. Primera combinación de datos

Respecto a las combinaciones con atributos categoriales se tiene que para la primera existe una mejor distribución de instancias predichas respecto a su contraparte para el modelo SVM, en donde se agrupan las predicciones en torno a los porcentajes 0 %, 10 %, 20 %, 50 % y 100 %. En el modelo GB se puede ver una mayor frecuencia de intentos de clasificación para los porcentajes positivos en general, como se puede

ver en la Figura 7.8. En el resto de las ejecuciones para este dominio se nota un comportamiento similar, aunque no tan preciso como en la primera. A nivel colectivo se mantiene la tendencia a duplicar la medida de *ranking* respecto a los atributos normalizados, como sucede en el modelo SVM.

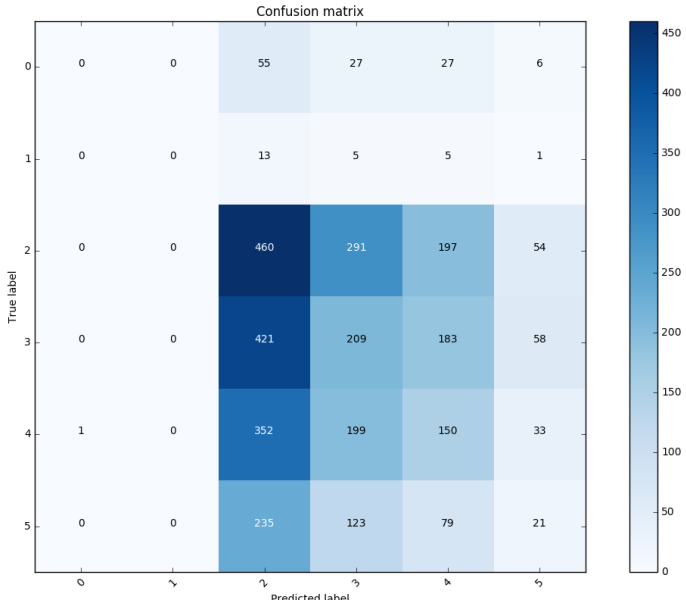

Figura 7.9: Modelo GB, atributos normalizados y clases según calidad. Primera combinación de datos

Luego, para las combinaciones de la distribución según **calidad** se tiene un resultado similar a la distribución anterior; y del mismo modo que se presenta para el modelo SVM de clasificación, a mayor proporción de acierto individual, mayor es el *ranking* colectivo. Además, en la Figura 7.9 se puede ver una mejor agrupación de los valores predichos por el clasificador respecto a lo obtenido para el modelo SVM. En este modelo, las características que representan respuestas malas y muy malas no son aprendidas por los clasificadores. Si bien esta es una observación compartida con el modelo SVM, se nota que en GB directamente no se intentó predecir este tipo de respuestas.

Para atributos categoriales los valores de las ejecuciones se mantienen muy similares a los obtenidos para el modelo SVM, en el que respuestas *malas* y *muy malas* no logran destacarse, mientras que características de respuestas *excelentes* son aprendidas con mayor exactitud. Se observa una mejor distribución de respuestas clasificadas correctamente, lo que se ve reflejado tanto a nivel individual como colectivo. Las tres combinaciones presentan distribuciones semejantes, por lo que en la Figura 7.10 se puede ver la tercera combinación con mayor detalle, que es la que obtuvo mejores resultados. A diferencia de las ejecuciones con atributos normales se nota un intento de predecir respuestas muy malas, lo cual resulta en una clasificación incorrecta ya que los resultados verdaderos pertenecen a respuestas que no aportan o que apenas tienen algún punto.

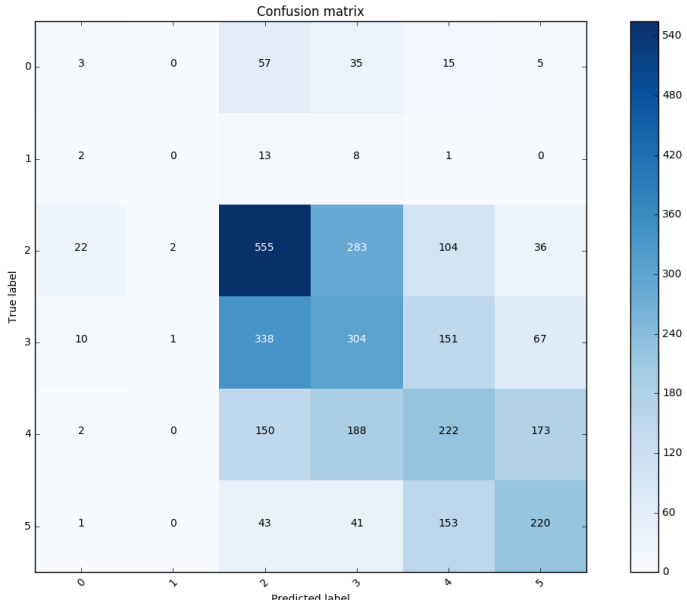

Figura 7.10: Modelo GB, atributos OHE, clases según calidad. Tercera combinación de datos

Por último, a diferencia de lo visto para el modelo SVM, para la distribución según **potencial** no se capturaron características ni prácticamente se intentó predecir la clase 0. A pesar de esto, SVM presenta mejores resultados tanto a nivel individual como colectivo. Comparando los valores obtenidos con los atributos categoriales, en la primera combinación de SVM parece haber una leve mejoría en la distribución al clasificar más respuestas como potenciales en lugar de decir incorrectamente que son buenas. Para la segunda combinación en GB sucede algo similar; dado que las respuestas *buenas* tienen mejores resultados que en la primera combinación.

Regresión

Los modelos GB de regresión son entrenados con los parámetros especificados en la Tabla F.8, ubicada en el Anexo F.

	Combinación 1			Combinación 2			Combinación 3		
Distrib.	**MSE**	**R2**	**Rank.**	**MSE**	**R2**	**Rank.**	**MSE**	**R2**	**Rank.**
Porcentajes	1300.83	-0.004	50.0 %	1300.68	-0.004	**50.0** %	1300.56	-0.004	50.0 %
Calidad	1.46	-0.003	**51.0** %	1.46	0.0001	49.0 %	1.46	-0.003	49.0 %
Potencial	0.33	-0.004	**50.0** %	0.33	-0.002	49.0 %	0.33	-0.003	49.0 %

Tabla 7.8: Resultados por clase para GB de regresión con atributos normalizados

	Combinación 1			Combinación 2			Combinación 3		
Distrib.	**MSE**	**R2**	**Rank.**	**MSE**	**R2**	**Rank.**	**MSE**	**R2**	**Rank.**
Porcentajes	852.92	0.341	**76.0** %	846.59	0.346	75.0 %	879.38	0.321	74.0 %
Calidad	0.90	0.380	74.0 %	0.88	0.395	**74.0** %	0.92	0.367	73.0 %
Potencial	0.23	0.301	68.0 %	0.22	0.315	**68.0** %	0.24	0.282	67.0 %

Tabla 7.9: Resultados por clase para GB de regresión con *one hot encoding*

Respecto a la distribución según **porcentaje**, se observa un comportamiento similar al modelo anterior: se tiene un MSE grande, producto del tamaño del dominio de la clase objetivo, una relación inversamente proporcional entre el *ranking* y MSE, y se conserva la relación del porcentaje de puntuación entre las respuestas a su respectiva pregunta.

Una diferencia importante a remarcar respecto al modelo anterior es que para esta clase —particularmente con OHE— el MSE disminuyó considerablemente respecto al modelo anterior. Se presume que esto se debe a la naturaleza del modelo, que básicamente consiste en reducir el MSE mediante la evaluación consecutiva de regresores débiles —como se menciona en la Sección 2.3—, por lo tanto es de esperarse que el modelo cuente con un MSE bajo.

Para la distribución según **calidad**, nuevamente producto del tamaño del dominio se observa un MSE menor que en la distribución anterior. Respecto al modelo anterior se observa una disminución en el MSE; esto como bien se menciona en la distribución anterior, se presume que es debido a la naturaleza del modelo.

Por último, para la distribución según **potencial** se observa, al igual que en las distribuciones anteriores, que se tiene una disminución del MSE debido al nuevo tamaño del dominio (tres valores), sin embargo y a diferencia del modelo anterior, el MSE disminuye más en proporción con respecto a la distribución anterior. Como bien se marca en el modelo anterior, al utilizar OHE se obtienen mejores resultados debido a la representación que se obtiene de los atributos de categoría. Al igual que en el modelo anterior, la métrica R2 reafirma esta idea ya que se cuenta con valores positivos de esta para las combinaciones con OHE y valores negativos para casi todas las combinaciones con atributos normalizados.

Relevancia de atributos

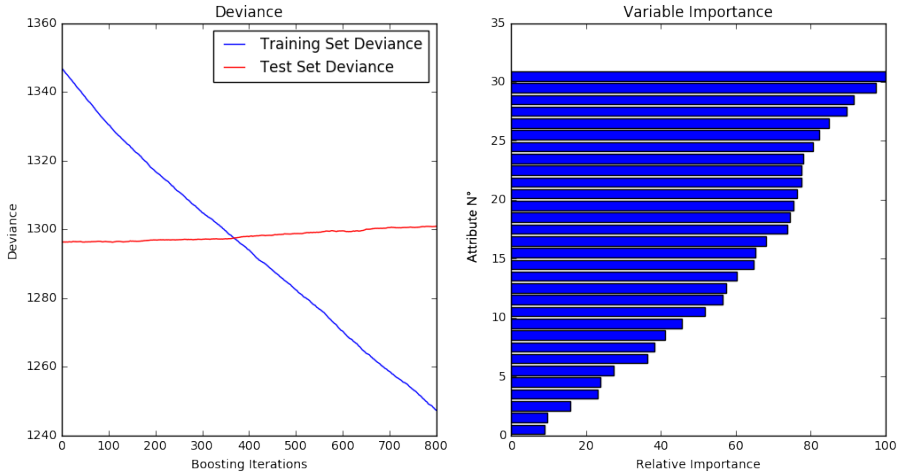

Figura 7.11: Desviación estándar y relevancia de atributos de una combinación de datos al azar

En la Figura 7.11 se muestra el análisis del modelo GB para una combinación

de datos al azar; la gráfica izquierda representa la desviación estándar al evaluarlo en el conjunto de entrenamiento y la desviación estándar al evaluarlo GB en el conjunto de evaluación, para cada iteración del modelo GB. Esto permite determinar la iteración en la cual el modelo empieza a sobreajustar los datos, lo cual ocurre en el momento que ambas desviaciones empiezan a alejarse, decreciendo la desviación en el conjunto de entrenamiento y aumentando levemente en el conjunto de evaluación. Según la Figura 7.11, el modelo GB empieza a sobreajustar luego de la iteración 400 (aproximadamente).

La gráfica de la derecha en la Figura 7.11 muestra los primeros 30 atributos ordenados ascendentemente según su relevancia en el modelo entrenado. Vale la pena observar que se cuenta con una gran cantidad de atributos y ninguno sobresale del resto, lo cual implica que GB deba realizar un análisis más exhaustivo de los atributos para mejorar el regresor, y esto se traduce en una cantidad elevada de iteraciones, como se puede observar en la primera gráfica (casi 400 iteraciones antes de alcanzar el mejor regresor).

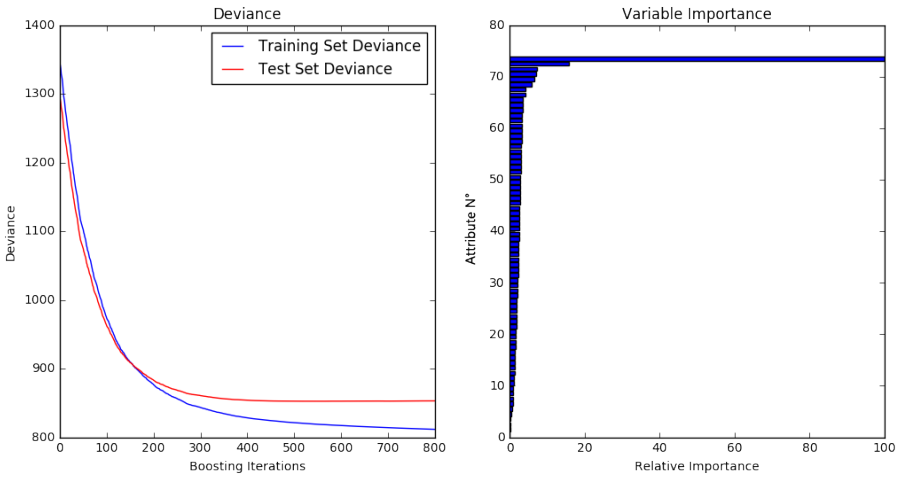

Figura 7.12: Desviación estándar y relevancia de atributos del mejor exponente

En la Figura 7.12 se muestra el análisis de GB para el mejor exponente. En la gráfica de relevancia por atributo se observa que existe un atributo que sobresale del resto, lo cual se traduce en menor número de iteraciones —con respecto al análisis anterior— antes de alcanzar el mejor regresor (menos de 200 iteraciones). Esto es un claro ejemplo de cómo una buena representación de los atributos, como lo es la generación de atributos de calidad y cantidad y OHE, influyen de forma sustancial en el desempeño del regresor.

7.5. Multilayer Perceptrons

A continuación se describen las ejecuciones de los modelos de clasificación y regresión de *perceptrones multicapa* (MLP) utilizados para predecir el porcentaje de los puntos totales de cada respuesta perteneciente a una pregunta, así como su orden relativo respecto a este atributo.

Clasificación

Los modelos MLP de clasificación son entrenados con los parámetros especificados en la Tabla F.9, ubicada en el Anexo F.

Durante las primeras pruebas con el modelo MLP, tanto para el modelo de clasificación como de regresión se debieron ajustar los parámetros de ejecución hasta obtener un resultado aceptable con el cual realizar las pruebas, dado que en primera instancia el modelo solamente fue capaz de predecir una sola clase, independientemente de la distribución de valores que tuviera el atributo objetivo. Luego de diversas pruebas, los parámetros de mayor relevancia a la hora de lograr una clasificación más diversa son: la cantidad de capas ocultas —*hidden_layer_sizes*—, la función de activación —*activation*— y la función que ajusta los pesos en la red —*solver*—. Variando estos atributos se tuvo la combinación inicial que permite realizar el análisis presentado a continuación, que se puede ver en la Tabla F.9, ubicada en el Anexo F.

	Combinación 1		**Combinación 2**		**Combinación 3**	
Clase objetivo	**Acc.**	**Rank.**	**Acc.**	**Rank.**	**Acc.**	**Rank.**
Porcentajes	15.66 %	43.0 %	15.91 %	**44.0** %	15.75 %	42.0 %
Calidad	25.30 %	40.0 %	24.83 %	**45.0** %	24.86 %	42.0 %
Potencial	45.80 %	48.0 %	46.27 %	48.0 %	46.67 %	**49.0** %

Tabla 7.10: Resultados por clase para una red neuronal de clasificación, con atributos normalizados

	Combinación 1		**Combinación 2**		**Combinación 3**	
Clase objetivo	**Acc.**	**Rank.**	**Acc.**	**Rank.**	**Acc.**	**Rank.**
Porcentajes	24.86 %	67.0 %	24.33 %	**68.0** %	22.40 %	66.0 %
Calidad	40.00 %	**66.0** %	38.75 %	65.0 %	37.50 %	62.0 %
Potencial	66.77 %	**62.0** %	65.43 %	61.0 %	63.40 %	59.0 %

Tabla 7.11: Resultados por clase para una red neuronal de clasificación con *one hot encoding*

A nivel general, tras analizar las matrices de confusión y la distribución de las predicciones obtenidas del clasificador se puede observar que para la distribución según **porcentajes** se tiene una leve mejoría en la variedad de las clases positivas predichas. Para el resto de las ejecuciones se presenta una similitud muy alta con los resultados obtenidos en el modelo GB, por lo cual no se cree necesario visualizar las figuras en este apartado. No obstante, todas las figuras se encuentran en el tercer apartado del Anexo F, correspondiente a los resultados de los modelos MLP.

Para la distribución según **porcentajes** y atributos normalizados se puede observar que a nivel individual se presentan los resultados con menor rendimiento de

los tres modelos analizados; sin embargo ocurre lo opuesto para la medida de *ranking*, alcanzándose un máximo de 44 % para la segunda combinación, casi un 10 % más que para los otros modelos. Es posible que la razón de este comportamiento se deba a lo que se comenta en el párrafo anterior: al presentar una mayor variedad de resultados, inclusive al no ser exactos —lo cual explica la decaída a nivel de *accuracy*—, a la hora de analizar el orden de las respuestas es factible que se haya encontrado un orden más flexible que en los otros modelos.

En la misma línea que lo dicho en el párrafo anterior, para la distribución según **calidad** se reflejan resultados individuales levemente inferiores que los obtenidos con los otros modelos, aunque se nota que las predicciones —aunque incorrectas— presentan una mayor variedad, lo que probablemente resulte en una mejora de la medida de *ranking*. En cuanto a los atributos con dominio categorial, y las combinaciones para la distribución según **potencial**, los resultados son prácticamente idénticos a los obtenidos para el modelo GB. En el Anexo F se pueden encontrar con mayor detalle los resultados individuales.

Regresión

Los modelos MLP de regresión son entrenados con los parámetros especificados en la Tabla F.10, ubicada en el Anexo F.

	Combinación 1			**Combinación 2**			**Combinación 3**		
Distrib.	**MSE**	**R2**	**Rank.**	**MSE**	**R2**	**Rank.**	**MSE**	**R2**	**Rank.**
Porcentajes	1336.36	-0.031	**51.0** %	2246.86	-0.734	51.0 %	2270.29	-0.752	51.0 %
Calidad	1.45	0.001	**52.0** %	1.47	-0.005	48.0 %	1.46	-0.003	49.0 %
Potencial	0.33	-0.001	**51.0** %	0.33	-0.005	50.0 %	0.33	-0.007	50.0 %

Tabla 7.12: Resultados por clase para una red neuronal de regresión, con atributos normalizados

	Combinación 1			**Combinación 2**			**Combinación 3**		
Distrib.	**MSE**	**R2**	**Rank.**	**MSE**	**R2**	**Rank.**	**MSE**	**R2**	**Rank.**
Porcentajes	1039.53	0.197	**74.0** %	1169.81	0.096	72.0 %	1117.59	0.137	72.0 %
Calidad	0.91	0.375	**74.0** %	1.28	0.126	70.0 %	1.40	0.039	70.0 %
Potencial	0.23	0.286	**68.0** %	0.23	0.290	67.0 %	0.24	0.264	66.0 %

Tabla 7.13: Resultados por clase para una red neuronal de regresión con *one hot encoding*

Para la distribución según **porcentajes**, como en los modelos anteriores se vuelve a observar un MSE elevado, producto del tamaño del dominio de la clase objetivo, para el cual se corresponde un porcentaje de *ranking* similar a los modelos anteriores. Luego, para la distribución según **calidad** nuevamente y como ocurre con los modelos anteriores, el MSE disminuye con respecto a la clase anterior. Por último, para la distribución según **potencial** no se tienen grandes variaciones con respecto a los modelos anteriores.

Una vez más se observa un decremento del MSE debido al nuevo tamaño del dominio de la clase objetivo, y a su vez nuevamente se obtienen mejores resultados al utilizar OHE, debido a su representación de los atributos de categoría. Al igual que en el modelo anterior, la medida R2 reafirma esta idea ya que se cuenta con valores positivos de esta para las combinaciones con OHE y valores negativos para casi todas las combinaciones con atributos normalizados.

7.6. Mejores resultados

A continuación, se exponen a modo de resumen los mejores resultados obtenidos para cada modelo. El criterio utilizado para selección de mejor clasificador o regresor se basa en obtener una buena relación entre las predicciones individuales y las colectivas. Por esta razón, predomina el valor de *accuracy* para los modelos de clasficación y el puntaje R2 para los de regresión; en caso de desempate se opta por el que tenga mayor valor de *ranking*. Se observa con interés el resultado obtenido tras realizar el ordenamiento interno ya que es lo que se busca como objetivo primario de la solución, pero se le da prioridad a la medida objetiva de cada tipo de problema, ya que a la hora de trabajar sobre los parámetros del modelo lo que se optimiza es justamente este valor. Además, conceptualmente no son tipos de problemas con medidas compatibles a la hora de evaluar el rendimiento de los modelos; la medida *ranking* se ve como una forma de poder tener un punto de comparación más objetivo entre los tipos de problemas.

Distrib. objetivo	**Dominio**	**Modelo**	**Ejec.**	**Accuracy**	**Ranking**
Porcentajes	Normalizado	GB	2	21.93 %	34.0 %
	One hot encoding	MLP	2	24.86 %	67.0 %
Calidad	Normalizado	GB	3	27.70 %	41.0 %
	One hot encoding	GB	3	40.68 %	66.0 %
Potencial	Normalizado	GB	1	48.95 %	48.0 %
	One hot encoding	GB	2	67.36 %	63.0 %

Tabla 7.14: Mejores exponentes por clase para modelos de clasificación

Si bien en líneas generales los resultados obtenidos son muy similares, para los modelos clasificadores existe un predominio del modelo GB, y solo en una ocasión un modelo MLP logró imponerse como mejor modelo, como se puede ver en la Tabla 7.14. Por el lado de los modelos de regresión, se comparte el predominio del modelo GB, seguido en dos instancias por un modelo MLP, como se puede ver en la Tabla 7.15. Se observa que para las tres distribuciones del atributo objetivo se destaca la predominancia del dominio de atributos transformados a *one hot encoding* tanto para los modelos clasificadores como para los regresores. Existe un salto notorio en la medida de *ranking* obtenida para este dominio, en el que se logran valores que oscilan entre el 60 % y 76 %, mientras que para los atributos normalizados estos valores oscilan entre un 30 % y un 48 %, lo cual sugiere que el hecho de controlar y simplificar los valores a procesar por el modelo produce una mejora sustancial al predecir las instancias.

Por otro lado se puede notar que a medida que se reduce la cantidad de clases disponibles, el porcentaje de *ranking* disminuye. Esto es razonable si se toma en cuenta que a nivel individual cada modelo no es lo suficientemente preciso como para poder discriminar exactamente entre las respuestas, y dado que se reduce el espectro de opciones posibles es esperable que muchos valores sean iguales —en el caso de la clasificación— o muy cercanos —en el caso de la regresión—, lo que al ordenar las respuestas aumenta la probabilidad de error. Para el caso de los modelos clasificadores, el hecho de que se agrupe la gran mayoría de las instancias alrededor de un porcentaje de puntos de la respuesta cercano al cero —identificadas también como respuestas que *no aportan*— hace que los valores predichos se repitan, y no se pueda establecer un criterio mejor al ordenar las respuestas. Cuando se toma la distribución según **porcentajes**, al existir más valores posibles aumenta la posibilidad de que aunque no se clasifique correctamente la pregunta a nivel individual, a nivel colectivo se logre mantener el orden.

Otra observación esperable es que en modelos de regresión a medida que baja el valor de MSE suele aumentar el valor de *ranking*. Esto puede deberse al hecho de que cuando baja el MSE el error medio es cada vez menor, el modelo es más preciso y por lo tanto al predecir valores reales se puede establecer el orden de mejor manera, lo cual da lugar a mejores resultados respecto a los obtenidos en los modelos de clasificación.

No es posible afirmar nada respecto a los atributos utilizados en las distintas ejecuciones debido a que para los mejores modelos no existe una predominancia en cuanto a una mejor ejecución. De todas formas, en caso de adherirse a los mejores resultados en general expuestos por los modelos regresores, se tiene que en tres situaciones se tuvo como mejor combinación la número uno, que utiliza los atributos de cantidad y calidad entre respuestas y sus preguntas, mientras que en dos casos se utilizaron los atributos originales. Por último y con el porcentaje de *ranking* más bajo en este tipo de modelos se tuvo la combinación número tres, que hace referencia a los atributos sin información de usuario. Es interesante notar que, más allá de que exista una cierta tendencia hacia los atributos de la primera combinación, no se obtuvieron grandes diferencias entre los atributos utilizados, lo cual hace interesante el planteo de utilizar estos modelos con datos que no cuenten con información de usuario.

De los mejores resultados obtenidos para cada dominio y distribución, se elige el obtenido por el modelo **Gradient Boosting** de **regresión**, para los atributos con ***one hot encoding*** aplicado, distribución según **porcentajes** y atributos de calidad y cantidad entre las respuestas y sus preguntas, resaltado en la Tabla 7.15.

Para este modelo se realiza un entrenamiento tomando la distribución original del atributo objetivo —sin redondear a la decena más cercana— y se obtienen algunos ejemplos de respuestas bien ordenadas, con el fin de observar los puntos altos del modelo. Las métricas obtenidas se detallan en la Tabla 7.16.

Distrib. objetivo	Dominio	Modelo	Ejec.	MSE	R2	Ranking
Porcentajes	Normalizado	GB	3	1300.56	-0.004	50.0 %
	One hot encoding	GB	1	852.92	0.341	76.0 %
Calidad	Normalizado	MLP	1	1.45	0.0001	52.0 %
	One hot encoding	GB	2	0.88	0.395	74.0 %
Potencial	Normalizado	MLP	1	0.33	-0.001	51.0 %
	One hot encoding	GB	2	0.22	0.315	68.0 %

Tabla 7.15: Mejores exponentes por clase para modelos de regresión

Distrib. objetivo	Dominio	Modelo	Ejec.	MSE	R2	Ranking
Original	One hot encoding	GB	1	843.46	0.3383	75.0 %

Tabla 7.16: Mejor exponente con distribución original de atributo objetivo

Instancia de ejemplo

Un caso en que se obtuvo un ordenamiento correcto se puede ver en la Tabla 7.17. En ella se puede notar que los valores predichos se acercan a sus valores originales y se mantiene el ordenamiento original, aunque no necesariamente se haya acertado en su totalidad a la predicción individual. Para el tipo de problemas que se quiere resolver, si bien es importante una correcta aproximación individual al porcentaje obtenido por una respuesta, cobra mayor relevancia un correcto orden, que es lo que el potencial usuario llegaría a ver en un caso de uso real. Al entrar a sitios como Stack Exchange en busca de una respuesta, normalmente el usuario se queda con lo que ve primero, en este caso, las respuestas con mejor porcentaje de puntos recibidos.

Respuesta	Puntaje	Porcentaje original	Porcentaje predicho
1	11	52.0 %	62.5 %
2	8	38.0 %	20.0 %
3	2	9.0 %	3.8 %
4	-1	-100.0 %	-3.6 %

Tabla 7.17: Predicciones de porcentaje de puntos de una respuesta en el modelo para la pregunta con Id 3.544

El título y cuerpo de la pregunta del ejemplo mostrado en la Tabla 7.17, junto con sus respuestas, puede encontrarse en las Tablas 7.18 y 7.19. La temática de la pregunta seleccionada refleja la duda sobre el correcto uso de dos palabras similares; la mejor respuesta plantea una definición concreta, con enlace externo a un diccionario, una comparación con Google y un par de sugerencias. La segunda y tercera respuesta —en orden de relevancia— brindan una opinión más personal, citando un ejemplo de un diccionario. Por último, la cuarta respuesta consiste de una definición extremadamente larga y posiblemente irrelevante para quien creó la pregunta.

Título	"Undistinguishable" vs. "indistinguishable"
Cuerpo	Is there a difference between these two words? To me, it seems that undistinguishable is more where you can't tell what it is, and indistinguishable seems to be where they're the same. It seems a lot of places list them as synonyms though.

Tabla 7.18: Título y cuerpo de la pregunta 3.544.

En el sexto apartado del Anexo F se encuentra una tabla que contiene los atributos utilizados para clasificar las respuestas mencionadas. Si bien los datos presentan similitudes, hay casos particulares donde se notan diferencias y se cree que el modelo traza la línea para discriminar los porcentajes de las respuestas. En la Tabla 7.20 se muestra un resumen de estos atributos. La primera respuesta posee el porcentaje más alto, lo que es respaldado por el atributo *q_ accepted_ answer*, que indica la respuesta seleccionada por el usuario si existe. Se observa que frecuentemente este porcentaje se relaciona con el atributo de probabilidad de selección de respuesta que tiene el usuario que la creó, ya que se indica que hay una probabilidad alta pues sus respuestas tienden a ser seleccionadas como mejores.

Por otra parte, los atributos con mayor variabilidad en el modelo son los calculados a partir de las relaciones entre cada respuesta y su pregunta. En la mejor respuesta se destaca un promedio de longitud de la respuesta elevado —aunque para la peor respuesta también se tiene alto este valor—, mientras que la peor respuesta tuvo una alta cantidad de pronombres y un mayor valor en el atributo calculado de *calidad*. De estos atributos no se logran obtener conclusiones claras; si bien muchas veces se observan datos interesantes para definir la calidad de una respuesta, en ocasiones estos resultan contradictorios por lo que no terminan resultando de utilidad. En la Tabla 7.20 se observa que se favorecen las respuestas más antiguas, lo que sugiere que la peor respuesta puede estar relacionada con una poca cantidad de puntos debido a su cercanía en el tiempo, más allá de que la respuesta en sí sea mala. A nivel del atributo de similitud coseno, la tercera y cuarta respuesta presentan valores más cercanos a la pregunta original, y se piensa que esto ocurre porque en las respuestas se mencionan explícitamente y con mayor frecuencia palabras encontradas en la pregunta.

Es de interés investigar cuáles fueron los atributos que predominaron al momento de establecer un correcto ordenamiento interno. Para esto se toman todas las preguntas cuyas respuestas fueron correctamente ordenadas y se analizan los valores que tomaron los atributos descriptos en el sexto apartado del Anexo F. En la Figura 7.13 se puede un ejemplo donde para los valores posibles de un determinado atributo —cantidad de entidades nombradas promedio entre una respuesta y su pregunta—- se agrupan las instancias de acuerdo al valor de porcentaje objetivo. En este ejemplo puede verse que para la clasificación aprendida por el modelo terminan prevaleciendo los valores bajos —que exhiben una mayor concentración de instancias— por sobre los altos de este atributo.

En el sexto apartado del Anexo F se pueden ver las gráficas para cada uno de los atributos analizados. Además del atributo mencionado de entidades nombradas, se tiene que para la cantidad de sustantivos también se agrupa en la zona baja, mien-

Respuesta 1 - Puntaje: 11	I'm a native English speaker, and I've never heard of "undistinguishable". I searched for undistinguishable and Google replied with: "Did you mean: indistinguishable" Princeton University's WordNet defines indistinguishable as: - identical: exactly alike; incapable of being perceived as different; "rows of identical houses" "cars identical except for their license plates" "they wore indistinguishable hats" - not capable of being distinguished or differentiated; "the two specimens are actually different from each other but the differences are almost indistinguishable"; "the twins were indistinguishable"; "a colorless person quite indistinguishable from the colorless mass of humanity" To convey the sense of "you can't tell what it is", you could use indecipherable or inscrutable.
Respuesta 2 - Puntaje: 8	I've never seen "undistinguishable" before. My spell-check flags it as an error and suggests "indistinguishable". I suspect it's a typo or a case of misspelling a word in a logical way. I can't imagine that its meaning would be different from ïndistinguishable". The Corpus of Contemporary American English (COCA) lists only 8 hits for "undistinguishable" and 1000+ hits for "indistinguishable". I'd stick to the latter.
Respuesta 3 - Puntaje: 2	"Undistinguishable" may perhaps be used only regionally now, I have heard it a lot in my life, but I am from western North Carolina, where Standard English is rarely spoken. It may not be part of contemporary Standard English. Etymonline has an entry for it, listing it from the 1580s meaning 'not distinguishable'.
Respuesta 4 - Puntaje: -1	http://www.englishforums.com/English/PrefixesNegativePrefixes/wvkl/post.htm Look for Califjim's answer. You will be impressed! I pasted it below for your convinience. There is no rule. Words with these prefixes have come about through accidents of history. The most usual is "un-", but always consult a dictionary. The following does not really answer your question, but you may find it somewhat useful anyway, especially if you're willing to work to dig some of this out of a dictionary. Dissertation on "Negative Prefixes" in English. "a-" is a Greek prefix meaning "not" or "without". It is found almost exclusively with words formed from Greek roots. You can usually spot these by the spellings: "ph", "th", "y", "rh", "chr", "pn", "mn", final "sis" or "ic". theist / atheist chromatic / achomatic rhythmic / arhythmic symmetry / asymmetry This prefix is found mostly in scientific terminology, especially in the medical sciences. "agranulocytosis", "apnea", "amenorrhea", "anemia", "apraxia", "amitosis". However, these are not cases where the prefix was applied to an already existing word. Most people know these words as a single unit. They are unaware that the initial "a" has a separate meaning of its own. These should be learned separately, as there are very few pairs like those cited above. This prefix is also confusable with the native English prefix "a-", as in "ago", "asleep", "aside", which does not have anything to do with negation. ...

Tabla 7.19: Cuerpo de respuestas a la pregunta 3.544

tras que para el resto de los atributos calculados —definidos en la Sección 6.2— se puede ver una distribución más pareja. Para el atributo de cantidad de días de una respuesta predominan las respuestas antiguas, lo que sugiere que el modelo no maneja tan bien las respuestas más nuevas —en este caso, una cantidad menor a seis meses contabilizada desde setiembre de 2014 hacia atrás—. Para la reputación del usuario no se tienen instancias de reputación baja o muy baja, lo cual sugiere que el modelo favorece las instancias de reputación media a alta. Algo similar ocurre con las cantidades de medallas: dentro de las instancias clasificadas correctamente por completo, no se encuentran casos de usuarios con pocas cantidades de medallas en general.

Para el atributo de calidad de una respuesta no se encuentran instancias con calidad *muy alta*. Para las medidas de distancia entre pregunta y sus respuestas no se puede apreciar una agrupación hacia los valores más cercanos, por lo que se asume que estos atributos no son determinantes a la hora de predecir. En la mayor parte de las gráficas se encuentran pocas instancias negativas —prácticamente solo aquellas con un porcentaje de -100—, lo que puede ser reflejo de que si hay más de una respuesta con puntos negativos el modelo no sea capaz de discriminar correctamente.

Atributo	Respuesta 1	Respuesta 2	Respuesta 3	Respuesta 4
ans id	3547	3548	63182	147376
ans percentage score	38	52	9	-100
ans/q noun count	*high*	*high*	*low*	*high*
ans/q pronoun count	*low*	*low*	*low*	*high*
ans/q sentence average length	*low*	*high*	*low*	*high*
ans/q sentences count	*low*	*high*	*low*	*high*
ans/q verb count	*low*	*high*	*low*	*high*
ans/q word average length	*high*	*high*	*low*	*low*
ans/q word count	*low*	*high*	*low*	*high*
ans creation date days	*greater than year*	*greater than year*	*greater than year*	*between 6 months and an year*
ans quality	*low*	*low*	*low*	*high*
q accepted answer	*false*	*true*	*false*	*false*
q ans cos sim	*close*	*close*	*closest*	*closest*
q creation date days	*greater than year*	*greater than year*	*greater than year*	*greater than year*
user ans selected prob	*average*	*many*	*few*	*few*

Tabla 7.20: Resumen de atributos relevantes para la pregunta con Id. 3544

7.7. Optimizaciones

A efectos de optimizar los parámetros utilizados durante la ejecución y dada la imposibilidad de ajustar todos los modelos utilizados debido al tiempo necesario de ejecución, se selecciona el mejor modelo entrenado y se le realiza un ajuste de hiperparámetros en búsqueda de mejorar los resultados obtenidos.

Ajuste de hiperparámetros

Debido a que se particiona el conjunto de instancias disponibles en datos de entrenamiento, validación y validación de hiperparámetros, no son necesarias técnicas como validación cruzada para evaluar el desempeño del modelo entrenado. Cuando se entrenan los modelos, como se menciona en la Sección 6.2, se separa de las instancias totales un 10 % —aproximadamente 6.000 instancias— para evaluación, y a su vez de ese conjunto se separa la mitad para validar la optimización de hiperparámetros, como se muestra en la siguiente sección.

Para la optimización de hiperparámetros se selecciona el modelo GB de regresión de la primera combinación, con distribución según decena más cercana al porcentaje de respuestas y dominio de atributos con *one hot encoding* aplicado. Para este modelo se tuvo un valor de *ranking* de un **76 %** de acierto, lo cual indica que de un conjunto de respuestas para una pregunta, se estima en este porcentaje la probabilidad de correctitud de su orden interno.

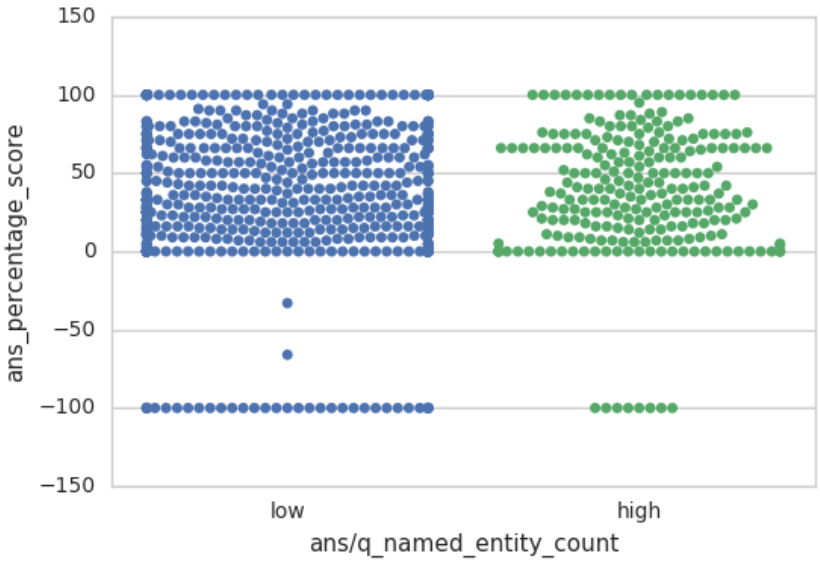

Figura 7.13: Distribución de valores categoriales para el atributo de cantidad de entidades nombradas entre pregunta y respuesta

Al desarrollar las primeras pruebas con la clase Gradient Boosting Regressor de Scikit-learn se realizó un ajuste básico de parámetros —detallado en la Tabla F.11 ubicada en el Anexo F— para lograr resultados base sobre los cuales trabajar. Sobre esta base de parámetros se utiliza la clase provista por Scikit-learn *RandomizedSearchCV*,[1] que permite entrenar un modelo una cantidad definida de veces, cada una con una combinación única de parámetros. Estos se especifican como un diccionario, donde para cada parámetro se define una lista de valores posibles.

En la Tabla F.11 ubicada en el Anexo F, se pueden ver los valores de atributos definidos para que la clase *RandomizedSearchCV* realice una serie de 50 iteraciones, cada una aplicando validación cruzada tres veces, lo que da una suma final de 150 ejecuciones.

En la Tabla 7.21 se encuentran los tres mejores resultados tras la aplicación del ajuste de hiperparámetros. Se aprecia que para el conjunto de datos a entrenar los parámetros *learning_rate* y *max_depth* se mantuvieron constantes en los mejores resultados, variando levemente el resto de los parámetros. Según lo investigado y comentado en la Sección 2.3, se confirma la preferencia de árboles más simples —con menor profundidad—.

Tras los resultados de la ejecución de *RandomizedSearchCV*, se entrena un modelo GB de regresión con los mejores parámetros obtenidos. Los resultados obtenidos se reflejan en la Tabla 7.22, donde el modelo entrenado es evaluado con un conjunto de pruebas reservado para esta ocasión con la misma cantidad de preguntas que el conjunto evaluado en las Tablas 7.14 y 7.15 .

[1] http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html

Parámetro	**Resultado 1**	**Resultado 2**	**Resultado 3**
learning_rate	0.01	0.01	0.01
max_depth	6	6	6
max_features	0.2	0.1	0.2
min_samples_leaf	15	12	8
n_estimators	500	1000	800
Puntaje *R2*	0.375	0.375	0.375
Desv. estándar	0.002	0.003	0.003

Tabla 7.21: Resultados tras la aplicación de ajuste de hiperparámetros

Distrib. objetivo	**Dominio**	**Modelo**	**Ejec.**	**MSE**	**R2**	**Ranking**
Porcentajes	One hot encoding	GB	1	817.58	0.375	76.0 %

Tabla 7.22: Modelo entrenado luego del ajuste de hiperparámetros

Tras comparar los resultados, se puede ver que para el modelo ajustado se tiene una mayor medida de puntaje R2, y a su vez decae levemente el valor del MSE. Esto indica que a nivel individual se logró una pequeña mejora, pero a nivel colectivo se mantuvo el puntaje original.

7.8. Comparación con otros resultados

Tras evaluar los resultados obtenidos de la ejecución, se propone una breve comparación con otros modelos o soluciones similares de manera de estudiar qué tan conveniente es formular este tipo de problemas desde un punto de vista de aprendizaje automático.

Aunque en la Sección 3.1 se expone que la idea detrás del presente trabajo está basada en la investigación propuesta por SemEval en el año 2015 [23] —en particular la tarea 3—, se cree que realizar una comparación no aporta información sustancial sobre el desempeño de la solución ya que además de que el conjunto de datos utilizados es diferente, el objetivo principal es poder establecer un ordenamiento claro de las respuestas, y los datos provistos por SemEval no tienen información relevante como para realizarlo. En los datos extraídos de Stack Exchange el corte para evaluar esto es claro y granular —comparando el puntaje de la respuesta y la proporción de ese puntaje considerando el total de las respuestas—, mientras que los datos de SemEval son clasificados subjetivamente mediante evaluación humana, además de no proveer categorías que permitan establecer un orden entre las respuestas.

Por esta razón, se propone realizar un breve estudio de cinco atributos relevantes dentro de las instancias disponibles para la solución, en el que se tomen las preguntas y sus respuestas y se reordenen según el criterio definido en la Sección 7.2. Se define el orden de mejor respuesta de acuerdo a la **reputación del usuario** que la creó; el **largo de la respuesta**; el **total de medallas** del usuario y la **cantidad**

de palabras de la respuesta. En la Tabla 7.23 se comparan las ejecuciones con el mejor resultado obtenido en la solución desarrollada.

Modelo	**% en *ranking***
Solución desarrollada	76.0 %
Reputación del usuario	62.7 %
Total de medallas del usuario	59.1 %
Largo de la respuesta	58.0 %
Cantidad de palabras de la respuesta	57.9 %

Tabla 7.23: Comparación de solución con otros criterios

En la Tabla 7.23 se pueden apreciar los resultados obtenidos tras comparar la mejor ejecución de la solución —luego del ajuste de hiperparámetros— con estos modelos. Se puede ver que el mejor exponente de la solución desarrollada presenta una mejoría de al menos un 10 % frente a los modelos contrastados.

Con el fin de observar qué tan relevantes pueden ser tanto los atributos de usuario como los de la respuesta en cuanto a un ordenamiento basado solo en atributos particulares, se tiene que colectivamente los atributos de usuario evaluados en la Tabla 7.23 presentaron un mejor resultado frente a los atributos sintácticos de la respuesta. Un resultado de casi un 63 % de precisión al ordenar una lista de respuestas según la reputación de sus usuarios o de un 59 % tomando en cuenta solo las medallas del usuario sugieren que puede haber una fuerte relación entre la información que aporta un usuario y el orden de las respuestas; sin embargo, luego de lo analizado en esta sección se puede afirmar que para los modelos entrenados no se encontró una diferencia substancial en el uso o la ausencia de estos datos. Una de las razones de que se pierda esta relación puede estar en la cantidad de atributos con los que se entrena, ya que pueden estar introduciendo ruido al no aportar información de valor.

Capítulo 8

Conclusiones y trabajo a futuro

En este capítulo se presentan las conclusiones extraídas tanto de los resultados presentados en el Capítulo 7 como de la solución a que se aspiraba inicialmente. Se puede afirmar que se logró cumplir los objetivos marcados en el Capítulo 1, donde se planteó la construcción de un sistema capaz de predecir de manera objetiva la calidad de una respuesta dentro de un conjunto de respuestas. Luego, se delinea un posible trabajo a futuro.

8.1. Conclusiones

En la Sección 1.3 se presentó como objetivo principal el estudio de un problema de ordenamiento de respuestas en un sitio de Q&A como Stack Exchange de manera de poder predecir qué tan buena es una respuesta para luego efectuar su ordenamiento, de manera de llegar al mismo criterio logrado por la interacción de usuarios de Stack Exchange.

La solución del problema planteado se centra en estudiar las posibilidades que surgen al analizar los componentes que reflejan de manera general las características que hacen importante o no a un conjunto de palabras en determinado contexto. A la hora de aplicar métodos de aprendizaje automático a problemas reales, la principal limitación se centra en la dificultad de llegar a descubrir cuáles son las características que pueden hacer viable el aprendizaje de un modelo, y si bien elegir el modelo adecuado y sus parámetros de configuración es un componente importante, lo estudiado para dar una solución a este problema permite confirmar que es primordial una precisa selección de datos a estudiar.

Tras lo expuesto en la Sección 7.2 se puede concluir que, en cuanto a los modelos de predicción elegidos para estudiar la problemática expuesta, el modelo de Gradient Boosting de regresión es el que presenta mejor desempeño a la hora de seleccionar características que permitan ordenar lo más exactamente posible conjuntos de respuestas asociadas a preguntas dadas.

A nivel de atributos utilizados para el entrenamiento del modelo, luego de tres agrupaciones de datos en las cuales se probaron relaciones directas entre preguntas y sus respuestas, atributos de manera individual y la ausencia de información de usuario, se puede afirmar que no se encontró una diferencia sustancial a la hora de variar

estos conjuntos, aunque se presentó cierta tendencia a privilegiar la información que tiene un análisis puntual en la relación de distintos atributos entre la pregunta y sus respuestas, como se plantea en la Sección 6.2, y además posee información de usuario.

El desempeño de la solución se vio positivamente afectado al momento de trabajar con datos categoriales, mediante la aplicación de la técnica *one hot encoding*. Esta redefinición del dominio de los atributos logró mejorar los resultados obtenidos con los atributos normalizados de manera sustancial, lo que sugiere que el hecho de limitar los valores que el modelo puede utilizar para aprender no solo facilita el trabajo a la hora de comprenderlos, sino que le permite agrupar de mejor manera las características aprendidas. Si bien la determinación de estos atributos categoriales se realizó de forma subjetiva, queda como trabajo pendiente estudiarlos con mejor criterio, preferentemente de manera objetiva, con el fin de evaluar si una categorización más detallada produce mejores resultados.

Otro elemento que influyó notoriamente en el desempeño de los modelos fue la utilización de modelos de regresión en comparación con los de clasificación, especialmente a la hora de calcular el ordenamiento de las respuestas. Al no haber obtenido un buen puntaje de clasificación individual, se dio una tendencia a agrupar las instancias predichas hacia valores fijos, por lo que al ordenar la lista de respuestas se encontraron valores superpuestos, cosa que influyó negativamente en el orden final. Este comportamiento se evitó con el uso de modelos de regresión, ya que al devolver valores reales se facilitó la tarea de ordenamiento.

La cantidad de los datos obtenidos de SemEval no presenta un problema mayor, como se comprueba en la segunda parte del Anexo E, sino que su distribución —en su gran mayoría respuestas que aportan poco o nada a la pregunta— dificulta el aprendizaje balanceado de los modelos. Esto, sumado a una pobre capacidad de predicción a nivel individual, motivó el cambio de dominio a nivel del atributo objetivo, planteado para lograr captar características más generales sacrificando detalles. Al evaluar los resultados con este nuevo dominio se notó que si bien a nivel de predicción individual se aumentó la tasa de acierto —esperable dado que representa una menor cantidad de clases— se perdió eficacia para ordenar las respuestas, por lo que se termina prefiriendo un modelo menos exacto a la hora de predecir los puntos de una respuesta pero con una mejor idea de su posición en la lista de respuestas. Es por esto que no realiza una comparación con la investigación planteada por SemEval en el año 2015 —Sección 3.1—, sino que se prefiere evaluar la solución de la forma más directa y granular posible mediante atributos directos.

A modo de resumen, el problema presentó un desafío importante a la hora de establecer criterios que reflejen características relevantes de una respuesta o pregunta. Se logró estudiar el problema y desarrollar una solución satisfactoria, tanto a nivel personal como frente a soluciones similares con las que se la comparó. El estudio de distintos modelos de aprendizaje automático permitió abstraerse de una idea particular y buscar distintas alternativas a los problemas que surgieron en el desarrollo, lo que involucró diversas herramientas y metodologías a la hora de construir los atributos utilizados. Qué tan bueno es un usuario en una comunidad o qué cantidad de respuestas tiene son datos concretos que reflejan una cierta importancia

a la hora de evaluar una respuesta, pero no puede saberse *a priori* si datos como el largo de una respuesta, cantidad de verbos u otros datos sintácticos guardan relación alguna, y es lo que se intentó averiguar en este proceso. Por otra parte, el estudio del problema desde un punto de vista semántico abre puertas interesantes, como el modelado de diccionarios de palabras con temática similar definido en la Sección 6.2, o el estudio de relaciones basado en información que se pueda extraer de árboles sintácticos definidos en la Sección 6.2, y estimula a reforzar lo aprendido hasta el momento. Si bien lo predicho con estas características no se destacó de manera clara en los modelos, se cree que un estudio de mayor profundidad puede proveer mejores herramientas a la hora de definir la calidad de una respuesta.

8.2. Trabajo a futuro

Distintos enfoques, herramientas e ideas no terminaron siendo ejecutados en el proceso hacia la solución desarrollada, ya sea por razones de tiempo o por exceder el alcance inicial planteado. A continuación se exponen brevemente conceptos que se deberían tener en cuenta en una próxima revisión del problema de ordenamiento de respuestas en un ámbito de Q&A.

En primer lugar, un enfoque interesante sería buscar palabras o frases recurrentes que denoten certeza por parte del usuario al responder una pregunta, por ejemplo, que incluya palabras específicas como *diccionarios*, *definición* o similares, así como el análisis de si sus palabras son positivas o negativas. Al hacer esto, se podrían llegar a descartar respuestas con finalidad de burla, reformulaciones de pregunta o comentarios que no aporten a la temática. También puede ser interesante encontrar una relación entre respuestas que contengan frases de tipo "en mi experiencia", que denoten una perspectiva personal en la respuesta. El peso y sentido que se le puede llegar a dar a esto depende mucho del subsitio con el que se esté trabajando. Por ejemplo, si se está en un sitio de relaciones laborales es muy factible que respuestas basadas en experiencias propias sean importantes, mientras que si se está en un sitio donde se manejan temas con cierta exactitud seguramente se prefieran respuestas más objetivas.

Luego, un atributo que quedó pendiente de analizar en profundidad es el correspondiente a las etiquetas que poseen las preguntas. Como se define en la Sección 5.2, cada pregunta en Stack Exchange cuenta con un conjunto de etiquetas que la enmarcan en una o varias categorías. Para el presente trabajo esta información fue extraída de Stack Exchange pero no se utilizó para generar ningún atributo. Se podrían utilizar estas etiquetas junto con algún procesamiento de la respuesta para determinar si esta se encuentra dentro de la categoría de la pregunta; con esta idea se podrían descartar respuestas que en principio sean comentarios a otra respuesta, o que sean erróneas.

Otro punto a ser tomado en cuenta para un futuro es un mejor aprovechamiento de los árboles sintácticos generados con la herramienta Freeling, mencionados en la primera parte del Anexo C. Mediante la utilización de los archivos XML generados se procesaron diversos atributos, como la cantidad de verbos, sustantivos o pronom-

bres de una respuesta o pregunta, pero un estudio en mayor profundidad de esta información generada podría añadir atributos semánticos a los modelos actualmente implementados.

Además de los puntos mencionados, si bien en el Anexo C se menciona la técnica de análisis de componentes principales (PCA), esta no se llegó a implementar sobre el conjunto de datos utilizados dado que en pruebas generales no se logró un mejor rendimiento, y además se perdía la capacidad de análisis de los atributos seleccionados, por lo que quedó fuera del alcance del proyecto. Una vez analizados los resultados finales obtenidos e investigados los atributos más relevantes al modelo, puede llegar a ser de interés aplicar esta técnica para eliminar posibles atributos irrelevantes.

Por otro lado, un análisis de la curva de Precision-Recall permite determinar el umbral que maximiza estas medidas. Una vez obtenido este umbral, se lo puede utilizar para entrenar nuevamente al modelo y obtener uno que maximice estas medidas. En la solución se llegó a generar una curva de Precisión-Recall para el modelo SVM, sin embargo no se profundizó en el análisis para obtener el umbral que la maximice.

En cuanto a las técnicas y herramientas utilizadas para resolver el problema planteado, se decidió acotar con cierto criterio el uso de bibliotecas y metodologías, principalmente para no extender demasiado el trabajo y sobrepasar el alcance inicial planteado, por lo que técnicas como Deep Learning o herramientas como TensorFlow, si bien al día de hoy están en auge, se descartaron y se decidió la utilización de Scikit-learn y métodos de aprendizaje automático más clásicos.

A la hora de evaluar el rendimiento de los modelos de aprendizaje utilizados, como se detalla en el Capítulo 7, se utilizó una medida propia llamada *ranking*. Con el fin de brindar un análisis con una medida estandarizada para el tipo de problema planteado en este trabajo y poder compararlos de manera objetiva con resultados de otros trabajos, se sugiere la utilización de la medida *normalized discounted cumulative gain* [50].

Una alternativa a la representación actual del atributo objetivo —que se detalla en la Sección 5.3— es plantear solo porcentajes positivos en lugar de considerar los negativos también, con lo cual se lograría reducir a la mitad la cantidad de clases a analizar. Una forma de realizar esto, sin perder la información de las respuestas de mala calidad, sería tomar la respuesta negativa con menor cantidad de puntos y sumarle esta cantidad a todas las respuestas de la pregunta. De esta forma, la peor respuesta quedaría con 0 puntos, por lo que todas tendrían un porcentaje entre 0 % y 100 %. Esta representación, si bien reduciría el rango de valores manejados por el modelo, presenta el problema de que no se tendría un porcentaje fijo de referencia para discriminar las respuestas malas, sino que este sería variable, por lo que las respuestas malas quedarían asociadas a los porcentajes bajos del total.

Respecto a los atributos utilizados, sería de gran valor poder obtener una representación más precisa de la calidad de un usuario. Si bien se utilizaron atributos

como la reputación o el total de medallas, sería interesante profundizar en la información disponible del usuario, como por ejemplo los tipos de medalla recibidos y las razones por las cuales se dan esas medallas. Más atributos de este tipo podrían llegar a tomar mayor relevancia en el modelo y podrían brindar información de calidad para la toma de mejores decisiones.

Si bien se limitó la cantidad de información utilizada —datos hasta setiembre de 2014—, en la segunda parte del Anexo E se estudió la importancia de la cantidad de datos utilizados para entrenar los modelos. Se concluyó que la adición de instancias con la misma distribución de datos actual no aporta valor sustancial, pero en el caso de poder obtener información que permita balancear la distribución actual se equilibraría el entrenamiento de los modelos y se podrían predecir las instancias con mayor exactitud.

Por último, vale la pena mencionar que la solución desarrollada no utiliza estructuras ni atributos específicos del idioma inglés, por lo que en el caso de obtenerse una cantidad adecuada de instancias en otro idioma estos datos podrían procesarse y evaluarse del mismo modo que los utilizados en esta ocasión.

Bibliografía

[1] Jurafsky, D. (2000). *Speech and Language Processing*. Pearson Education India.

[2] Chowdhury, G. G. (2003). Natural language processing. *Annual review of information science and technology*, *37*(1), 51-89.

[3] Sinclair, J. (2005). *Developing Linguistic Corpora: a Guide to Good Practice Corpus and Text Basic Principles*.

[4] Mitchell, T. M. (1997). *Machine Learning* (1.ª ed.). New York, NY, USA: McGraw-Hill, Inc.

[5] Michalewicz, Z. y Michalewicz, M. (1997). Evolutionary Algorithms. En *Fuzzy Evolutionary Computation, Chapter 2* (pp. 264-269). Kluwer Academic.

[6] Dorigo, M., Birattari, M. y Stützle, T. (2006). Ant Colony Optimization – Artificial Ants as a Computational Intelligence Technique. *IEEE Comput. Intell. MAG*, *1*, 28-39.

[7] Mohri, M., Rostamizadeh, A. y Talwalkar, A. (2012). *Foundations of machine learning*. MIT press.

[8] Mikolov, T., Chen, K., Corrado, G. y Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *ArXiv preprint ArXiv:1301.3781*.

[9] Dubiau, L. (2013). *Procesamiento de lenguaje natural en sistemas de análisis de sentimientos*. http://materias.fi.uba.ar/7500/Dubiau.pdf. Último acceso: 19-11-2017.

[10] Hall, M., Frank, E. y Witten, I. H. (2016). *The WEKA Workbench. Online Appendix for Data Mining: Practical Machine Learning Tools and Techniques*.

[11] Colin, A. (1996). *Building Decision Trees with the ID3 Algorithm*. Ed. Miller Freeman.

[12] Zhu, M. (2004). Recall, Precision and Average Precision. *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo*, *2*, 30.

[13] Atwood, J. y Spolsky, J. (2009a). *Stack Exchange*. http://stackexchange.com/tour/. Último acceso: 19-11-2017.

[14] Atwood, J. y Spolsky, J. (2009b). *Stack Overflow*. http://stackoverflow.com/. Último acceso: 19-11-2017.

[15] Deterding, S., Sicart, M., Nacke, L., O'Hara, K. y Dixon, D. (2011). Gamification. Using Game-design Elements in Non-gaming Contexts. En *CHI '11 Extended Abstracts on Human Factors in Computing Systems* (pp. 2425-2428). CHI EA '11. Vancouver, BC, Canada: ACM.

[16] Ravi, S., Panga, B., Rastogi, V. y Kumar, R. (2014). Great Question! Question Quality in Community Q&A. *Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media*, 426-435.

[17] Stack Exchange Community. (2014a). *Stack Exchange: User Reputation.* http://meta.stackexchange.com/questions/7237/how-does-reputation-work. Último acceso: 19-11-2017.

[18] Stack Exchange Community. (2014b). *Stack Exchange: Voting Rules.* https://goo.gl/lNgy24. Último acceso: 19-11-2017.

[19] Karan. (2014). *Knowledge API for Quora. (Code removed).* https://github.com/karan/Qnowledge. Último acceso: 19-11-2017.

[20] Quora. (2013). *Does a Quora API Exist.* https://www.quora.com/Does-a-Quora-API-exist. Último acceso: 19-11-2017.

[21] Shippy, S. (2012). *Does the Quora iPhone Application use a Quora API.* https://www.quora.com/Does-the-Quora-iPhone-application-use-a-Quora-API. Último acceso: 19-11-2017.

[22] Montoya, N. M. (2005). ¿Qué es el estado del arte? *Ciencia y tecnología para la salud visual y ocular, herramientas para investigar*, 73-75.

[23] SemEval. (2015). *SemEval 2015.* http://alt.qcri.org/semeval2015/task3. Último acceso: 19-11-2017.

[24] Moschitti, A., Nakov, P., Marquez, L., Magdy, W., Glass, J. y Randeree, B. (2015). Semeval-2015 task 3: Answer Selection in Community Question Answering. *SemEval-2015.*

[25] Nicosia, M., Filice, S., Barrón-Cedeño, A., Saleh, I., Mubarak, H., Gao, W., ... y Magdy, W. (2015, junio). QCRI: Answer Selection for Community Question Answering - Experiments for Arabic and English. En *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* (pp. 203-209). Denver, Colorado: Association for Computational Linguistics.

[26] Zhou, X., Hu, B., Chen, Q., Tang, B. y Wang, X. (2015). Answer Sequence Learning with Neural Networks for Answer Selection in Community Question Answering. *CoRR, abs/1506.06490.*

[27] Tran, Q. H., Tran, V., Vu, T., Nguyen, M. y Bao Pham, S. (2015, junio). JAIST: Combining Multiple Features for Answer Selection in Community Question Answering. En *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* (pp. 215-219). Denver, Colorado: Association for Computational Linguistics.

[28] Hou, Y., Tan, C., Wang, X., Zhang, Y., Xu, J. y Chen, Q. (2015, junio). HITSZ-ICRC: Exploiting Classification Approach for Answer Selection in Community Question Answering. En *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* (pp. 196-202). Denver, Colorado: Association for Computational Linguistics.

[29] Leech, G. y Wilson, A. (1996). *Recommendations for the Morphosyntactic Annotation of Corpora EAGLES Report.* EAG-TCWG-MAC/R.

[30] Mohamed, R., Ragab, M., Abdelnasser, H., M. El-Makky, N. y Torki, M. (2015, junio). Al-Bayan: A Knowledge-based System for Arabic Answer Selection. En *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* (pp. 226-230). Denver, Colorado: Association for Computational Linguistics.

[31] Belinkov, Y., Mohtarami, M., Cyphers, S. y Glass, J. (2015, junio). VectorSLU: A Continuous Word Vector Approach to Answer Selection in Community Question Answering Systems. En *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* (pp. 282-287). Denver, Colorado: Association for Computational Linguistics.

[32] Ho, T. y X, Y. (2015). *Predicting Answer Quality on Community Q&A Websites.*

[33] Liu, T.-Y. (2011). The Pointwise Approach. En *Learning to Rank for Information Retrieval* (pp. 33-47). Berlin, Heidelberg: Springer Berlin Heidelberg.

[34] Cao, Z., Qin, T., Liu, T., Tsai, M. y Li, H. (2007). Learning to Rank: from Pairwise Approach to Listwise Approach. En *Proceedings of the 24th International Conference on Machine Learning* (pp. 129-136). ACM.

[35] Belinkov, Y., Barrón-Cedeño, A. y Mubarak, H. (2015). Answer Selection in Arabic Community Question Answering: A Feature-rich Approach. En *Proceedings of the Second Workshop on Arabic Natural Language Processing* (pp. 183-190).

[36] Wang, X.-J., Tu, X., Feng, D. y Zhang, L. (2009). Ranking Community Answers by Modeling Question-Answer Relationships via Analogical Reasoning. En *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 179-186). ACM.

[37] Wei, T., Lu, Y., Chang, H., Zhou, Q. y Bao, X. (2015). A Semantic Approach for Text Clustering Using WordNet and Lexical Chains. *Expert Systems with Applications*, *42*(4), 2264-2275.

[38] Miller, G. A. (1995). WordNet: a Lexical Database for English. *Communications of the ACM*, *38*(11), 39-41.

[39] Lucjon. (2010). *Py-Stack Exchange - A Python Binding for the Stack Exchange API.* GitHub.

[40] Hoogeveen, D., Verspoor, K. M. y Baldwin, T. (2015). CQADupStack: A Benchmark Data Set for Community Question-Answering Research. En *Pro-*

ceedings of the 20th Australasian Document Computing Symposium (ADCS) (3:1-3:8). ADCS '15. Parramatta, NSW, Australia: ACM.

[41] Padró, L. y Stanilovsky, E. (2012, mayo). FreeLing 3.0: Towards Wider Multilinguality. En *Proceedings of the Language Resources and Evaluation Conference (LREC 2012)*. ELRA. Istanbul, Turkey.

[42] Loper, E. y Bird, S. (2002). NLTK: The Natural Language Toolkit. En *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1* (pp. 63-70). ETMTNLP '02. Philadelphia, Pennsylvania: Association for Computational Linguistics.

[43] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A. y Varoquaux, G. (2013). API Design for Machine Learning Software: Experiences from the Scikit-Learn Project. En *ECML PKDD Workshop: Languages for Data Mining and Machine Learning* (pp. 108-122).

[44] McKinney, W. (2010). Data Structures for Statistical Computing in Python. En S. van der Walt y J. Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 51-56).

[45] Singhal, A. (2001). Modern Information Retrieval: A Brief Overview. *IEEE Data Eng. Bull. 24*(4), 35-43.

[46] Blei, D. M. (2012). Probabilistic Topic Models. *Communications of the ACM*, *55*(4), 77-84.

[47] Řehůřek, R. y Sojka, P. (2010, mayo). Software Framework for Topic Modelling with Large Corpora. En *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* (pp. 45-50). http://is.muni.cz/publication/884893/en. Valletta, Malta: ELRA.

[48] Blei, D. M., Ng, A. Y. y Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, *3*(Jan), 993-1022.

[49] Ng, A. Y. y Jordan, M. I. (2002). On Discriminative VS. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes. En *Advances in Neural Information Processing Systems* (pp. 841-848).

[50] Wang, Y., Wang, L., Li, Y., He, D. y Liu, T.-Y. (2013). A Theoretical Analysis of NDCG Type Ranking Measures. En *Conference on Learning Theory* (pp. 25-54).

Glosario

Clasificador

Se denomina *clasificador* a todo a todo algoritmo de aprendizaje automático que consiste en predecir la clase a la que pertenece un elemento basándose en sus características.

Corpus

Conjunto de anotaciones que se utiliza para representar las características de un texto.

GB

Acrónimo del inglés Gradient Boosting, un algoritmo de aprendizaje automático supervisado cuyo entrenamiento consiste en construir un predictor fuerte en base a varios predictores débiles.

MLP

Acrónimo del inglés *Multilayer Perceptron*, un algoritmo de aprendizaje automático supervisado.

OHE

One hot encoding (OHE) es una técnica utilizada para transformar atributos, en un formato que sea fácil de procesar para algoritmos de clasificación y regresión.

PCA

El análisis de componentes principales (PCA por sus siglas en inglés) es una técnica utilizada para reducir las dimensiones de un conjunto de datos, minimizando la pérdida de información.

PLN

Sigla cuyo significado es Procesamiento de Lenguaje Natural.

POS

Acrónimo del inglés *part-of-speech*, consiste en el proceso de etiquetar en categorías gramaticales a cada una de las palabras de un texto.

Q&A

Del inglés *question-answering*, término que alude a los sitios que ofrecen herramientas para preguntas y respuestas colectivas.

Regresor

Se denomina *regresor* a todo algoritmo de aprendizaje automático que consiste en predecir valor continuo en base a las características de un elemento.

SemEval

API, del inglés *application programming interface*, es un conjunto de subrutinas, funciones y métodos para ser utilizado por un componente de *software* como una capa de abstracción, permitiendo el acceso a los datos de forma ordenada y concisa.

Stem

Es el proceso por el cual se reduce una palabra a su raíz.

SVC

Modelo de *support vector machine* de clasificación.

SVM

Acrónimo del inglés *support vector machine*, un algoritmo de aprendizaje automático supervisado, que se utiliza para el análisis de datos tanto en problemas de clasificación como regresión.

SVR

Modelo de *support vector machine* de regresión.

Topic model

En el área del aprendizaje automático y el procesamiento de lenguaje natural, se conoce como *topic model* a un tipo de modelo estadístico para descubrir los temas abstractos que aparecen en una colección de documentos.