Universidad Nacional de Córdoba

Facultad de Matemática, Astronomía, Física y Computación

Trabajo final de la Licenciatura en Ciencias de la Computación

Aprendizaje automático para clasificación de actos de habla ilocutivos en mensajes de foros para educación a distancia

Estudiante: Mariano Hernán Piatti

Director: Dr. Marcos Javier Gómez

Profesora Representante: Dra. Luciana Benotti

Córdoba, Argentina

17 de Diciembre de 2021

Esta obra está bajo una Licencia Creative Commons Atribución – No Comercial – Sin Obra Derivada 4.0 Internacional.

Resumen

En este trabajo se investiga una tarea propuesta por la empresa Ikumi, desarrolladora del sistema *Mumuki*: un sistema online para aprender a programar. Mumuki cuenta con un foro donde los estudiantes consultan dudas sobre ejercicios de programación. En cursos multitudinarios, donde realizar un seguimiento personalizado es complejo, un foro de consultas puede sumar más trabajo de seguimiento al docente. Ikumi planteó la necesidad de ayudar al docente a clasificar los mensajes del foro. El objetivo de este trabajo es comparar modelos de aprendizaje automático que predicen si un mensaje escrito por un estudiante en el foro es un forward looking act o un backward looking act de acuerdo a la teoría lingüística de actos de habla. En esta teoría, un forward looking act es un mensaje que requiere una respuesta u otro tipo de reacción del interlocutor (por ejemplo, una pregunta, una orden, etc). Un forward looking act impone una obligación en el interlocutor que, si no se cumple disminuye la sensación de colaboración en la conversación. Un backward looking act es un mensaje que satisface una obligación conversacional y que no genera otra obligación (por ejemplo, una respuesta a una pregunta como "sí", un agradecimiento, un saludo de despedida). Para lograr el objetivo del trabajo, se realizó un análisis y anotación del conjunto de datos, luego se propusieron y entrenaron diversos modelos de aprendizaje automático midiendo el desempeño de la tarea propuesta, incluyendo una red neuronal recurrente en esta misma tarea. Como resultado de esta tesis se obtienen modelos construidos en base a datos generados por Mumuki con valores F1 por encima de 0.9 que son capaces de clasificar cada mensaje en tiempos de respuesta bajos, lo que permitirá que la integración al sistema Mumuki sea posible en tiempo real.

Abstract

This work investigates a task proposed by the Ikumi company, which develops the *Mumuki* system: an online system to learn how to program. Mumuki has a forum where students ask questions about programming exercises. In massive courses, where doing personalized follow-up is a complex task, a forum can add more follow-up work to teachers. Ikumi raised the need to help teachers classify forum messages. The objective of this work is to compare machine learning models that predict if a message written in the forum by a student is a forward looking act or backward looking act according to speech acts in linguistic theory. In this theory, a forward looking act is a message that requires an answer or other type of reaction from the interlocutor (e.g: a question, an order, etc). A forward looking act imposes an obligation on the interlocutor that, if it is not satisfied, hurts the collaborative feeling of the conversation. A backward looking act is a message that satisfies a conversation obligation and does not generate another obligation (e.g: an answer to a question like "yes", an acknowledgement, a farewell). To achieve the objective of this work, a dataset analysis and annotation was done, then different machine learning models were proposed and trained, measuring the performance of the proposed task. The result of this thesis are models built based on data generated in the Mumuki system with F1 values over 0.9 that can classify each message in low response time, which will allow the integration with the Mumuki system in real time.

Índice general

Capítulo 1

Introducción

1.1. Descripción de la Tesis

Detectar si un mensaje enviado por un estudiante en un foro educativo necesita una respuesta de un docente es una tarea que precisa de un seguimiento constante de los foros por parte de los docentes que utilicen los mismos. En los últimos años, debido al crecimiento de los cursos en línea, y potenciados por la virtualización de contenidos educativos por la pandemia, se han multiplicado los usuarios de estos sistemas tanto en el sistema educativo formal como no formal. Por lo tanto, podría ser útil para un docente contar con una herramienta que permita detectar de forma automática si un mensaje requiere una respuesta inmediata por parte del docente o se trata de mensajes en los cuales los estudiantes están interactuando entre ellos, agradeciendo o comentando sobre mensajes previos. Este trabajo final investiga[1] la tarea de modelar, usando aprendizaje supervisado y teoría de actos de habla, si un mensaje de un estudiante requiere una respuesta de un docente, dependiendo del mensaje que envía en un foro donde se debaten ejercicios para aprender programación. En particular, en esta tesis creamos modelos, a través de aprendizaje automático, que predicen si un mensaje de un estudiante es un acto de habla **forward** o **backward looking act**. A continuación explicaremos qué significan estos conceptos desde la perspectiva de la teoría de actos de habla, pero primero introduciremos brevemente el sistema de recolección de los datos usados. A lo largo de este trabajo, se trabajará sobre el sistema **Mumuki**[2] [5]. Mumuki es un sistema online para resolver ejercicios de programación utilizado por colegios, universidades, empresas y gobiernos. Mumuki cuenta con una versión abierta a autodidactas con miles de ejercicios para aprender programación de manera gratuita. Mumuki organiza la enseñanza en programas de estudio, los cuales

[1] El código fuente del trabajo de esta tesis está disponible en https://github.com/mariano-piatti/thesis
[2] https://mumuki.io

cuentan con un conjunto definido de ejercicios a resolver organizados en capítulos de temas particulares. Cada programa de estudio cuenta con un foro donde los estudiantes pueden realizar una consulta con respecto a un ejercicio en particular, lo que genera una conversación donde cualquier docente (a los que dentro de Mumuki llamaremos **mentores**) o estudiante puede participar. En estos foros a veces los estudiantes comentan o socializan intercambiando mensajes que no requieren una respuesta de un mentor. A veces los mentores deben dedicar una cantidad considerable de tiempo a leer cada mensaje del foro, en busca de un mensaje de algún estudiante que exprese explícita o implícitamente que necesita una respuesta, filtrando aquellos mensajes que no requieren ser respondidos. Una posible herramienta que ayude al mentor a priorizar su atención en aquellos estudiantes que necesiten respuestas, es clasificar de manera automática los mensajes como: si necesitan respuesta o no. Se utiliza en este trabajo un conjunto de datos proveniente de Mumuki, el cual cuenta con 9566 mensajes enviados en el foro de un programa de estudio sobre *Gobstones* [28], un lenguaje de programación pensado para la enseñanza introductoria a la programación.

La teoría de actos de habla nos dice que un acto de habla es una acción que el participante de una conversación realiza mediante algo que dice. A su vez, el **acto de habla ilocutivo** es la **intención** que tiene el participante de la conversación al decir algo. Si lo que dice tiene como intención introducir una obligación conversacional, entonces es un **forward looking act** o acto de habla que mira hacia adelante. Los forward looking acts introducen actitudes sociales que pueden imponer una obligación a ser atendida por el interlocutor, como por ejemplo pedir, ordenar o preguntar. En cambio, si la declaración realizada no introduce una obligación conversacional, es un acto de habla **backward looking act** o acto de habla que mira hacia atrás. Estos actos suelen ser reacciones a actos de habla previos y satisfacen una obligación conversacional, como por ejemplo responder, agradecer o felicitar. De esta manera, según la teoría de acto de habla, si un mensaje de un estudiante es un forward looking act, implica que el mensaje requiere atención y reacción de un mentor; mientras que si el mensaje es un backward looking act, no la requiere.

1.2. Motivación

Con el avance de la tecnología digital, el aprendizaje virtual a través de cursos en línea masivos y abiertos (también conocidos como **MOOC** [1] del inglés *Massive Open Online Course*) es cada vez más común en todo el mundo, lo que lleva a que cada vez más estudiantes tengan acceso a educación en línea.

En este trabajo nos enfocamos en Mumuki, la cual es una de las múltiples herra-

mientas que existen para practicar programación hoy en día. Mumuki es gratuita, de código libre y cuenta con más 2000 enunciados de ejercicios de programación en 17 lenguajes de programación distintos, más de 50 mil usuarios y más de 3 millones de soluciones enviadas en Argentina, lo cual muestra la gran demanda que hay de aprender a programar. Por más que este trabajo tiene foco en Mumuki, no es un caso aislado en nuestro país. La iniciativa de enseñar y aprender programación está incentivada a través de millonarias inversiones públicas e innumerables iniciativas en países como Estados Unidos, Inglaterra, Alemania, y también en Argentina con el proyecto Program.AR del Ministerio de Ciencia y Tecnología de la Nación [34, 8]. En el trabajo de Benotti et al. [5] se compara el aprendizaje del lenguaje de programación Haskell entre cursos que utilizan Mumuki y cursos donde no lo utilizan. En particular, en [5] también se detalla el funcionamiento de Mumuki y la retroalimentación (en inglés: *feedback*) automática que este da a la hora de enviar una solución de un ejercicio, el cual funciona como corrector automático del ejercicio. A pesar de la existencia de correcciones automáticas, el estudiante debe tener la posibilidad de acceder a la ayuda personalizada de un docente o de otros estudiantes, y aquí es donde entran en juego los foros de consultas.

Es común que un curso disponible en un MOOC cuente con un foro donde los estudiantes publican consultas con respecto a un ejercicio en particular del curso y los docentes las resuelven, e incluso otros estudiantes participan de la consulta, generando una conversación. El problema principal surge de la masividad de los MOOC, el hecho de que cada vez haya más estudiantes por cada docente afecta la atención que este pueda darles. Este trabajo propone explorar la posibilidad de contar con un sistema que ayude al docente a identificar aquellos mensajes que necesiten respuesta. El objetivo es asistir al docente a procesar muchos mensajes en poco tiempo.

Trabajo previo muestra que una respuesta con largos plazos de espera disminuye la efectividad de interacción entre estudiante y docente. Una rápida respuesta fomenta a que los estudiantes expresen sus dudas e incluso mejora la calidad de las participaciones de los estudiantes [18]. Además evita posibles frustraciones y potencial deserción de los estudiantes por el tiempo en espera sin atención. Esto último se aborda en el trabajo de Moresi [32], donde utiliza datos provenientes de Mumuki para predecir si un estudiante va a ser capaz de corregir los errores de un programa por su cuenta o necesita ayuda de un docente analizando los programas escritos por el estudiante.

Una de las principales motivaciones para este trabajo es el desafío de poder trabajar con un conjunto de datos sobre los cuales no hay trabajo previo, en español de variante argentina interactuando con una PyME argentina, Ikumi, y poder construir un modelo para ser integrado en un sistema que es realmente utilizado en la actualidad y que

puede tener un impacto directo en los mentores y estudiantes de Mumuki.

1.3. Objetivo y plan de trabajo

El objetivo principal de esta tesis es explorar la tarea de entrenar modelos de aprendizaje automático creados a partir del conjunto de datos de mensajes de foros provisto por Mumuki. Estos modelos clasifican si un mensaje de un estudiante es un forward o backward looking act, implicando, según la teoría de actos de habla, que requieren reacción de un mentor o no, respectivamente. Para lograr este objetivo se plantean distintas tareas a llevar a cabo.

Tarea 1 Analizar propiedades de la conversación y los actos de habla [14] presentes en sus mensajes, los cuales darán un mayor entendimiento de los fenómenos existentes en la comunicación entre las personas. También será importante entender estas propiedades en un contexto educativo.

Tarea 2 Estudiar trabajos previos existentes en aplicación de aprendizaje automático en mensajes en foros educativos.

Tarea 3 Analizar el funcionamiento de Mumuki, el flujo desde que un estudiante intenta resolver un ejercicio hasta que publica una consulta en el foro. Analizar el conjunto de datos, su estructura y la información presente que pueda ser útil para entrenar modelos de aprendizaje automático.

Tarea 4 A partir del conjunto de datos disponible, anotar los mensajes enviados en el foro por estudiantes en forward o backward looking acts, realizar un análisis post-anotación y buscar posibles características útiles para el entrenamiento de modelos de aprendizaje automático.

Tarea 5 Comprender cómo la tarea de determinar si un mensaje de un estudiante es un forward o backward looking act se puede modelar como un problema de clasificación binaria de aprendizaje supervisado. Entender que esto es una sobresimplificación dado que un mensaje puede cumplir ambos roles a la vez.

Tarea 6 Analizar el desempeño de modelos de aprendizaje automático como regresión logística [33], bosque aleatorio [7] y red neuronal recurrente LSTM [49].

1.4. Estructura de la tesis

Una vez concluida la introducción, a continuación se detalla la estructura de los siguientes capítulos.

Capítulo 2: Propiedades de la conversación humana y trabajo previo Se analizan propiedades de la conversación entre humanos, su complejidad y trabajos previos de clasificación de actos de habla en diálogos educativos.

Capítulo 3: Mumuki Se describe el sistema Mumuki, el cual dará soporte a este trabajo, y el conjunto de datos provisto por el mismo.

Capítulo 4: Metodología Se detalla el proceso de anonimización y anotación de los mensajes de estudiantes enviados en el foro de consultas de Mumuki obtenidos a partir del conjunto de datos, junto con un análisis post-anotación del mismo.

Capítulo 5: Aprendizaje automático para clasificación de mensajes Se describe cómo la tarea de determinar si un mensaje de un estudiante es un forward o backward looking act puede ser modelada como un problema de aprendizaje automático.

Capítulo 6: Experimentos y resultados Se presentan los resultados de distintos experimentos que utilizan aprendizaje automático para modelar el problema.

Capítulo 7: Conclusiones y trabajo futuro Se presentan conclusiones y se describe el trabajo futuro que da lugar esta tesis. Por último, se detallan las consideraciones éticas del trabajo.

Capítulo 2

Propiedades de la conversación humana y trabajo previo

En este capítulo analizamos las propiedades de la conversación entre humanos y su complejidad. En la Sección 2.1 detallamos propiedades y características de la conversación humana, como la base conversacional, las subconversaciones, la estructura de la conversación, la inferencia y la implicatura. En la Sección 2.2 describimos el concepto de acto del habla y los clasificaremos en forward y backward looking act para distinguir aquellos mensajes que requieren la atención de un mentor de los que no. En la Sección 2.3 definimos los cursos en línea masivos (en inglés *MOOC: Massive Open Online Course*) y analizamos el trabajo previo clasificando actos de habla en diálogos educativos.

2.1. Conversación

La conversación es un acto natural en el que dos o más personas participan expresándose a través de un medio con el fin de comunicarse. El medio utilizado para conversar es comúnmente verbal, aunque también puede ser por escrito, como es el caso de las conversaciones que analizaremos a lo largo de este trabajo, o visual, como por ejemplo con lenguaje de señas. Es una de las primeras actividades que aprendemos a realizar de niños y es el tipo de lenguaje con el que la mayoría de los humanos más cómodo se siente. En la Figura 2.1 vemos un ejemplo de conversación extraído de [22].

En la sección a continuación introducimos con ejemplos los turnos en la conversación y definimos el concepto de terreno común.

2.1.1. Turnos y terreno común

Una conversación está organizada en **turnos** en los que cada participante contribuye a la conversación. En la Figura 2.1 hay 20 turnos. Cada turno va cambiando de uno a otro participante y en cada uno el mismo puede comunicar ya sea una frase, una sola palabra, o varias oraciones. La finalización del turno de un participante, sobre todo cuando la conversación es un diálogo de voz, es muy importante ya que generalmente se intenta evitar las superposiciones entre participantes o interrupciones. Un ejemplo de superposición se da en la conversación de la Figura 2.1 en A16-C17. Cuando se trata de una conversación por escrito, por ejemplo un chat, la finalización del turno es bastante explícita ya que está dada por la finalización de la escritura del mensaje que emite el participante.

C1: ...Necesito viajar en el mes de Mayo.
A2: Y en qué día de Mayo le gustaría viajar?
C3: Ok… Necesito estar allí para una reunión que se dará del 12 al 15 de Mayo.
A4: Y a qué ciudad desea volar?
C5: Seattle.
A6: Y a qué hora le gustaría partir de Pittsburgh?
C7: Mmm no creo que haya muchas opciones de vuelo directo.
A8: Cierto. Hay tres opciones de vuelo directo al día de hoy.
C9: ¿Cuáles son?
A10: La primera parte de PGH a las 10:00hs, y arriba a Seattle a las 12:05hs hora de allí. La segunda parte de PGH a las 17:55hs y arriba a Seattle a las 20:00hs. La última opción parte de PGH a las 20:15hs y arriba a Seattle a las 22:28hs.

C11: Ok, tomaré el vuelo de las 17:55hs la noche previa, el 11 de Mayo.
A12: ¿El 11 de Mayo? Ok. Parte a las 17:55hs arriba a Seattle a las 20hs. Número de vuelo 115.
C13: Ok.
A14: ¿Y además dijo que volvía el 15 de mayo?
C15: Mmm, si, al final del día.
A16: Ok. Hay # dos opciones de vuelo directo #
C17: # En… En realidad #, ¿Qué día de la semana cae el 15 de Mayo?
A18: Un viernes.
C19: Mmm, mejor voy a quedarme allí un día más, hasta el domingo.
A20: Ok, ok… El domingo están las opciones…

Figura 2.1: Fragmento de conversación entre un agente turístico (A) y un cliente (C), la parte enmarcada por # significan una superposición de ambos al hablar.

Una conversación no es simplemente una serie de actos de habla independiente, sino más bien un acto colectivo llevado a cabo por el hablante y el oyente. Como en

toda actividad colectiva, es importante para los participantes establecer en qué están de acuerdo, llamado **terreno común** (del inglés common ground). Los hablantes hacen esto basándose en las declaraciones de los otros, es decir, haciéndole saber al oyente que han entendido lo que han dicho previamente. Por ejemplo en la Figura 2.1, en A8 el agente de viajes comienza su turno con "*Cierto*" por lo que deja en claro que ha entendido lo que el cliente dijo antes. También se suele repetir lo que la otra persona acaba de decir, un ejemplo de esto es en A12, donde el agente de viajes comienza con "*¿El 11 de Mayo? Ok.*". Otro ejemplo de establecer una base se dan en los A2 Y A4 de la Figura 2.1 donde el agente comienza su declaración con "Y".

En la sección a continuación analizamos la estructura general de la conversación, los pares de adyacencia y se desarrolla el concepto de subconversación.

2.1.2. Estructura de la conversación y subconversaciones

Sacks [42] introduce el análisis conversacional, donde se dice que la conversación tiene una estructura local. Las preguntas dan lugar a una respuesta, las propuestas a una aceptación o rechazo, los cumplidos a un agradecimiento o minimización. A estos pares Schegloff [45] los llama **pares de adyacencia** y se componen por el primer turno y el segundo turno. En las conversaciones el primer turno no siempre va seguido de un segundo turno, ya que puede haber casos en los que se de una subconversación entre ellos. Por ejemplo en la Figura 2.1 de C7 a A10 se da una subconversación, C7 no contesta la pregunta A6 directamente. La pregunta A6 se contesta en C11, una vez que el cliente sabe cuáles son los horarios disponibles. En C17 hay una **subconversación de corrección**. El agente debe responder la pregunta de C17 y considerar que en C19 el cliente en realidad está interesado en un vuelo para dos días más tarde, el 17 de Mayo, y proceder a buscar vuelos de regreso para esa fecha. Otro tipo de subconversación es una **subconversación de aclaración**, que puede suceder entre una petición y una respuesta, por ejemplo, cuando el hablante hace una pregunta y el oyente no logra discernir una palabra porque un auto pasó por la calle haciendo mucho ruido. Entonces, se generaría una subconversación donde el oyente le pide a la otra persona que aclare la palabra.

En la sección que sigue introducimos la importancia de la inferencia por parte del oyente para la comprensión de una conversación, también definimos la implicatura y las máximas que existen en las conversaciones.

2.1.3. Inferencia e implicatura

La inferencia es una parte importante en la compresión de la conversación, por ejemplo en la Figura 2.1 en A2 y C3, el cliente no contesta la pregunta del agen-

te exactamente, solo menciona una reunión en un cierto tiempo. Grice [15] introduce ejemplos como estos en su teoría de **implicatura conversacional**, donde el hablante parece esperar que el oyente realice ciertas inferencias, es decir, que el hablante comunica mayor información de la que parece estar presente en la declaración realizada. La implicatura es una clase particular de inferencia. Grice propone que lo que hace que los oyentes realicen inferencias es que la conversación está guiada por **máximas**, que son heurísticas generales que juegan un rol de guía en la interpretación de declaraciones conversacionales. La **máxima de la relevancia**, sugiere que el hablante quiere ser relevante en lo que dice, y no está simplemente haciendo declaraciones aleatorias. En la Figura 2.1, en C3, el agente debe razonar que hay una cierta relevancia por la cual el cliente está mencionando la reunión, y teniendo en cuenta que generalmente cuando se da una reunión, la persona que asiste debe estar en el lugar, por lo que debe viajar hasta el lugar, y como a la gente le gusta llegar un día antes de la reunión, el agente debe inferir que el vuelo debe ser el 11 de Mayo.

Muchas de las inferencias conversacionales son necesarias para poder distinguir distintos actos de habla. Por ejemplo, para saber que C3 es una respuesta a A2 y no una simple aserción no relacionada es necesario hacer las inferencias que se describieron anteriormente. Para poder realizar estas inferencias es necesario contar con información del contexto extralingüístico de la conversación. En este ejemplo ese contexto incluye las costumbres de viajes de personas que van a reuniones. En nuestra tesis ese contexto estará dado por características de los estudiantes y de su progreso en el aprendizaje.

En la sección a continuación se introduce el concepto de acto de habla.

2.2. Actos de habla

En esta sección introduciremos el concepto de acto del habla, la importancia que tiene en la comunicación en una conversación, sus clasificaciones y el concepto de *forward y backward looking act*, que nos permitirá distinguir aquellos actos de habla que requieren de una atención por parte del oyente. Luego utilizaremos esta clasificación para poder distinguir qué mensajes de estudiantes requieren la atención de un mentor.

En la sección que sigue daremos la definición de acto del habla, sus tres tipos: locutivo, ilocutivo y perlocutivo. Luego, definiremos 4 clases de actos de habla ilocutivos.

2.2.1. Definición y descripción

Un **acto del habla** [14] es algún tipo de acción que el hablante lleva a cabo en la declaración que realiza dentro de una conversación. Este concepto fue originalmente

introducido por el filósofo Wittgenstein [51], pero ampliamente desarrollado por los filósofos John Austin [3] y John Searle [46]. En la Tabla 2.1 podemos ver las tres distinciones de tipo de acto del habla que Austin sugiere.

Acto locutivo	Es el hecho propio de realizar una declaración.
Acto ilocutivo	Es la intención que tiene el hablante al realizar una declaración.
Acto perlocutivo	Es el efecto que tiene la declaración en el oyente.

Tabla 2.1: Los tres tipos de actos del habla

El acto locutivo es equivalente a pronunciar una expresión con un cierto nivel de sentido y referencia. Ejemplos de actos locutivos pueden ser "*El día está caluroso*" ó "*La caja pesa 10 kilos*", donde en la primera el hablante hace referencia al estado del tiempo real del día y en la segunda el hablante se refiere al peso real y exacto de la caja. Un acto ilocutivo es un acto que se realiza a través de la fuerza comunicativa de la declaración, y esta fuerza es la que el hablante desea. También se lo llama el acto de hacer algo diciéndolo. Ejemplos de actos ilocutivos son prometer, disculpar, ofrecer, advertir, sugerir, felicitar, etc. Siguiendo los ejemplos dados anteriormente, en la declaración "*El día está caluroso*", se podría solicitar que se encienda un ventilador. "*La caja pesa 10 kilos*" podría ser un pedido de ayuda para cargar la caja. Un acto perlocutivo es el acto que lleva a cabo el hablante y se refiere al efecto que causa la declaración ya sea en los pensamientos o en las acciones del oyente. En los ejemplos dados, "*El día está caluroso*" podría convencer al oyente de que prenda un ventilador. "*La caja pesa 10 kilos*" podría hacer que el oyente ayude a cargar la caja.

Según Bach y Harnish [4] [16], podemos distinguir 4 clases de actos de habla ilocutivos. En la Tabla 2.2 podemos ver las 4 clases y su descripción.

Constatativos	Comprometen al hablante a que algo sea cierto.
Directivos	Intentos del hablante para que el oyente haga algo.
Comisivos	Comprometen al hablante a una acción futura.
Reconocimientos	Expresan la actitud del hablante con respecto al oyente, relacionado a alguna acción social.

Tabla 2.2: Las 4 clases de actos del habla ilocutivos

Los actos ilocutivos constatativos son las declaraciones que denotan la creencia del hablante y su intención o deseo de que el oyente tenga o forme una creencia

similar. Pueden ser declaraciones para responder, reclamar, confirmar, negar, estar en desacuerdo, declarar, etc. Los actos ilocutivos directivos son las declaraciones que denotan la intención del hablante de alguna acción futura del oyente y la idea de que su declaración o la actitud que expresa sea tomada como una razón para la acción del oyente. Pueden ser declaraciones para aconsejar, pedir, prohibir, invitar, ordenar, pedir, etc. Los actos ilocutivos comisivos denotan la intención y creencia del hablante de que su declaración lo obliga a sí mismo a hacer algo (posiblemente bajo ciertas circunstancias). Pueden ser declaraciones para prometer, comprometer, planificar, apostar, etc. Los actos ilocutivos de reconocimiento son expresiones de sentimientos del hablante con respecto al oyente, o, si es una declaración muy formal, puede denotar la intención del hablante de satisfacer una expectativa social de expresar sentimientos y su creencia de que lo hace. Pueden ser declaraciones para disculparse, saludar, agradecer, felicitar, aceptar un reconocimiento, etc.

En la sección que sigue definimos los conceptos forward y backward looking acts.

2.2.2. Forward y Backward looking acts

Los actos de habla pueden clasificarse de acuerdo a si generan o no una obligación conversacional en *forward looking acts* o *backward looking acts* [37]. Los forward looking acts introducen actitudes sociales que pueden imponer una obligación a ser atendidas por el oyente.

Generalmente los forward looking acts son los siguientes actos de habla:

- **Actos constatativos**
- **Actos directivos**
- **Actos comisivos**
- **Afirmaciones**: Declaraciones que buscan lograr que la creencia del oyente.
- **Actos que buscan influencia futura en las acciones del oyente**: Limitan la situación de la declaración para darle una opción al oyente.
- **Actos de opción abierta**: No es un intento de hacer que el oyente haga algo, sino que habilita una posibilidad a que lo considere.
- **Ofertas**: Es un compromiso condicional que el oyente puede aceptar o rechazar.

Los backward looking acts pueden ser respuestas a actos de habla previos. Por ejemplo, el hablante puede aceptar o rechazar una propuesta previa, o responder a una petición de información. Generalmente los backward looking acts son los siguientes actos de habla:

- **Aceptar una afirmación/petición/acuerdo total o parcialmente**
- **Rechazar una afirmación/petición/acuerdo total o parcialmente**
- **Posponer la respuesta a una afirmación/petición/acuerdo**
- **Responder**
- **Agradecer**
- **Felicitar**
- **Reclamar**

2.3. Detección de actos de habla en diálogos educativos

En esta sección analizamos la conversación en un contexto educativo, donde los participantes de la misma son estudiantes y mentores. Para este trabajo nos interesa analizar las conversaciones dadas en foros de consultas en el marco de plataformas para educación en línea y tener en cuenta el trabajo previo realizado en este campo.

En la Sección 2.3.1 definimos los cursos en línea masivos (en inglés *MOOC: Massive Open Online Course*), sus beneficios y detallamos los puntos críticos como un contexto donde tiene sentido la aplicación de detección automática de actos de habla. Por último, en la Sección 2.3.2 analizamos en profundidad tres trabajos previos en el área de clasificación de actos de habla en mensajes de foros educativos.

2.3.1. Ventajas y desafíos de los MOOC

Los MOOC [1] son una modalidad de estudio que consiste en cursos en línea disponibles para una gran cantidad de estudiantes en todo el mundo. Con la evolución de la tecnología e internet han tenido su auge en los últimos años, desarrollándose comúnmente en asociaciones cooperativas internacionales como por ejemplo Coursera[1], que es una asociacion de 62 universidades de todo el mundo dirigida por la Universidad de Stanford o edX[2], que incluye al Instituto de Tecnología de Massachusetts, la École Polytechnique Fédérale de Lausanne, la Universidad de Ciencia y Tecnología de Hong Kong. Otros ejemplos pueden ser Udemy[3] o Khan Academy[4]. Los MOOC generalmente agrupan gente interesada en aprender (estudiantes), y un experto o expertos interesados en facilitar el aprendizaje (mentores). La conectividad entre ellos se da a través de redes sociales que los MOOC proveen donde están

[1] https://www.coursera.org
[2] https://www.edx.org
[3] https://www.udemy.com
[4] https://es.khanacademy.org

disponibles materiales de estudio y espacios para compartir información, debatir, y consultar. Los MOOC tienen beneficios como:

- Acceso masivo de estudiantes interesados en un tema en particular.
- Pueden acceder estudiantes de todo el mundo gracias al desarrollo completo en línea.
- El acceso es abierto, libre y gratuito, generalmente no requiere conocimientos previos.
- Contenido abierto.
- Motiva el aprendizaje autónomo.
- Permite la interacción entre estudiantes y mentores.

A lo largo de este trabajo nos basaremos en el MOOC Mumuki [5]. Mumuki[5] es un sistema online de soporte para aprender a programar utilizado por colegios, universidades, empresas y gobiernos, sumando más de 150 mil personas aprendiendo a programar y ofreciendo también una versión abierta para particulares interesados en aprender programación. Cuenta con distintas guías con más de 2000 ejercicios en diferentes lenguajes de programación y un foro de consultas donde estudiantes y mentores debaten sobre distintos ejercicios.

Los MOOC también tienen sus **puntos críticos** como toda modalidad de aprendizaje, de los cuales podemos destacar tres.

La adopción del contenido no es tarea fácil

Primero, hay un desafío importante relacionado con la adopción, compresión y desarrollo de los contenidos presentes en un curso en línea, sumado a que muchas veces estos contenidos son costosos de realizar y luego no son fácilmente reutilizables en otros contextos educativos. En el trabajo de Brusilovsky et al. [8] explora este desafío. Se definen los términos contenido de aprendizaje electrónico (del inglés *electronic learning content* (ELC)) y contenido de aprendizaje inteligente (del inglés *smart learning content* (SLC)). El primero hace referencia a los tradicionales contenidos educativos estáticos como texto, imágenes o videos que son utilizados comúnmente en sistemas de administración del aprendizaje (como por ejemplo Moodle[6]). Ejemplos de ELC pueden ser Wikipedia o Youtube. SLC hace referencia a un tipo de contenido educativo que ha surgido en los últimos 10 años que deja atrás al contenido estático y proporciona una participación más interactiva e implementa mecanismos de retroalimentación

[5] https://mumuki.io/
[6] https://moodle.org

más complejos. Ejemplos de SLC en el área de enseñanza de la programación pueden ser entornos de programación, de simulación, de visualización de ejecución de programas, de resolución de problemas, etc. En [8] se aborda esta problemática enfocada en los SLC y se realiza una encuesta a listas de mail de la SIGCSE[7], de csed-research[8], de PPIG[9] y de Problets[10], solicitando a educadores de ciencias de la computación que seleccionen de una lista los desafíos con los que se habían encontrado a la hora de **elegir una SLC apropiada**. Como resultado se obtuvieron que las tres mayores dificultades fueron:

- Tuvieron problemas a la hora de elegir un SLC que puedan usar.
- No pudieron personalizar un SLC a la hora de adecuarla a las necesidades locales.
- No pudieron integrar un SLC con otros sistemas en su institución.

También, a la hora realizar la misma pregunta pero esta vez con respuesta abierta, obtuvieron que las mayores tres dificultades fueron:

- Dificultad técnicas a la hora de utilizar un SLC
- Aprender a utilizar un SLC requiere tiempo
- Poca disposición a utilizar un SLC hasta que su utilidad sea demostrada

El trabajo concluye que demasiado esfuerzo es puesto en desarrollar continuamente nuevo contenido, y propone metodologías para que estos contenidos sean reutilizables y adaptables a distintos estudiantes.

Masividad de estudiantes

Como segundo punto crítico de los MOOC, la **masividad de estudiantes** lleva a que haya una falta de ayuda y retroalimentación personalizada por parte de los mentores debido a la gran cantidad de consultas, lo que genera deserción. En el trabajo de Moresi [32] se aborda la deserción de los estudiantes mediante la propuesta de modelar automáticamente cuándo un alumno está en riesgo de abandonar un ejercicio y necesita una ayuda personalizada. Moresi trabaja sobre un conjunto de datos que consiste en códigos escritos en el lenguaje de programación Haskell dados por Mumuki, que muestra altas tasas de deserción como todos los MOOC. En la Figura 2.2 podemos ver el porcentaje de completitud de ejercicios en 5 proyectos distintos en Mumuki, por lo que podemos determinar los niveles de abandono a lo largo de cada proyecto, donde

[7]The ACM Special Interest Group on Computer Science Education - https://sigcse.org/sigcse/
[8]CS Education Research - https://csedresearch.org/
[9]Psychology of Programming Interest Group - https://www.ppig.org/
[10]http://problets.org/

cada uno cuenta con un conjunto de ejercicios a completar. Sobre el eje x tenemos el porcentaje de ejercicios completados y sobre el eje y tenemos el porcentaje de estudiantes inscriptos que completaron los ejercicios. Se puede ver como en el caso del proyecto 2 y 5 más del 50 % de los estudiantes inscriptos no lograron completar el 50 % de los ejercicios, mientras que para el proyecto 1 y 4 se da lo mismo en más del 40 % de los estudiantes inscriptos. Esto muestra que la mayor cantidad de abandonos en los proyectos de Mumuki se dan al principio del proyecto. A partir de ahí, se puede observar que la cantidad de estudiantes que completan el 50 % o el 75 % de los ejercicios y luego abandonan es más baja, en ninguno de los proyectos supera el 20 % de los estudiantes. Esto da la idea de que si un estudiante ya completó más del 50 % de los ejercicios, es probable que termine completando el 100 % de los ejercicios.

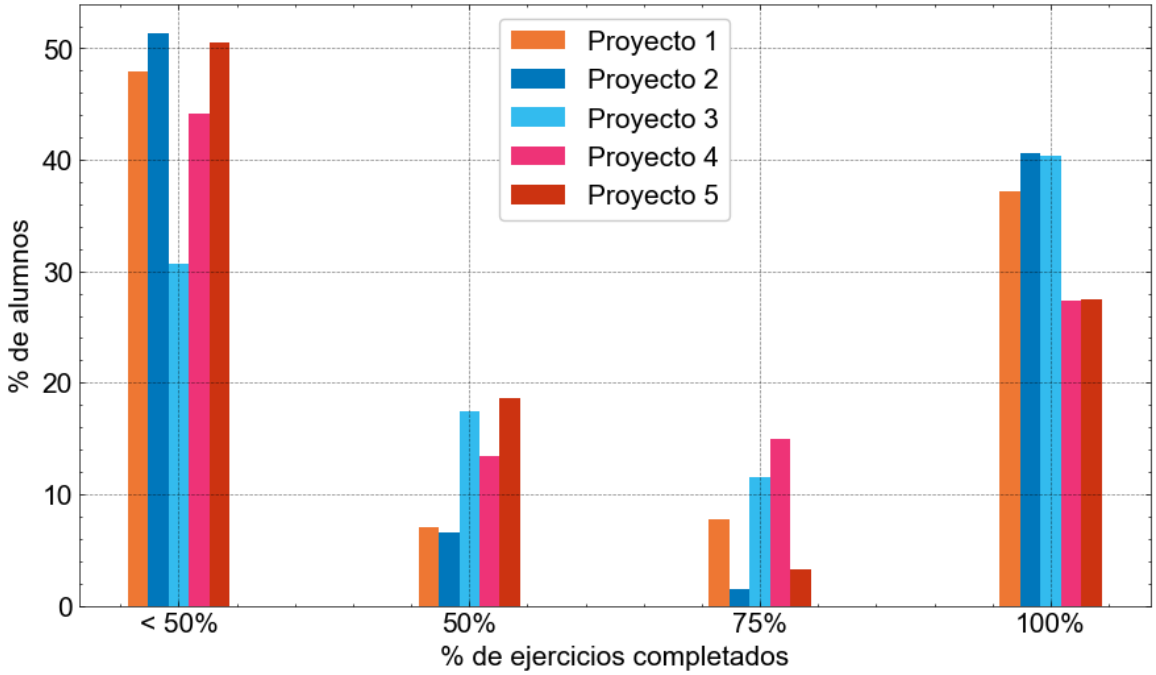


Figura 2.2: Porcentaje de ejercicios completados por estudiantes en 5 proyectos distintos de Mumuki.

Moresi en su trabajo, con el fin de detectar el abandono con anticipación, crea una red neuronal recurrente que toma como input el programa escrito en Haskell por el estudiante y determina si se producirá el abandono en ese mismo ejercicio o no comparando el texto de ese estudiante con un gran número de soluciones de otros estudiantes resolviendo ese ejercicio. Gómez en [12], en base a un programa escrito en Haskell intenta predecir el nivel de fluidez de un estudiante con respecto al lenguaje de programación. También, a partir de la masividad, en el trabajo de Irish et al. [19] investigan herramientas para ayudar en el manejo de grandes volúmenes de posts en

foros de consultas. Este gran volumen dificulta a los mentores poder responder, y a los estudiantes poder encontrar respuestas a consultas que probablemente ya hayan sido realizadas en el pasado. A partir de esto, Irish et al. crean una herramienta de recomendación llamada PARQR, que sugiere respuestas previas que pueden ser relevantes cuando los estudiantes escriben sus posts, evitando que se creen posts masivamente y disminuyendo la cantidad de posts duplicados. Otro acercamiento al desarrollo de herramientas para ayudar a los mentores a responder se da en el trabajo de Jenders et al. [20], en el cual se analizan discusiones en un MOOC llamado *openHPI*, que ofrece cursos de ciencias de la computación en inglés y alemán. En los foros de discusión generalmente el creador de la consulta selecciona la respuesta que le proporcionó la información que necesitaba como respuesta válida para que los demás estudiantes sepan cual es la respuesta correcta, pero a veces el creador de la consulta olvida hacerlo o lo omite. La idea en este trabajo fue construir un modelo que predijera cuál es la respuesta correcta basándose en un conjunto de datos de 835 preguntas con una respuesta marcada como correcta y 1841 respuestas en total. Jenders et al. probaron decidir cuál respuesta es la correcta mediante diferentes heurísticas base como por ejemplo elegir la primera respuesta, o la última respuesta, o la respuesta cuyo autor tenga más votos positivos en toda la plataforma, o más respuestas marcadas como correcta en toda la plataforma, o haya hecho más comentarios en toda la plataforma. Aunque es interesante tener estas heurísticas en cuenta, Jenders et al. logró mejores resultados utilizando técnicas como *Random Forest*, *Multilayer perceptron*, *Bagging* y *Naive Bayes*, tomando los mensajes de una discusión sumado a características del usuario que representan su actividad previa en el foro. Se consideran características de la conversación por ejemplo la cantidad de mensajes o información de tiempo entre respuestas y características con respecto al contenido del mensaje como por ejemplo la cantidad de palabras utilizadas o medidas de similitud entre la pregunta y la respuesta presente en el mensaje. Para predecir cuál es la respuesta correcta se observa cuál es la que tiene mayor puntaje de predicción, pudiendo tener aciertos, desaciertos y empates. El mejor resultado posible lo obtuvo la técnica Random Forest. Por otro lado, Rohloff [39] presenta el problema de que al ofrecer los MOOC la oportunidad de educación a miles de estudiantes con diferentes bases culturales y sociales, los mentores y el curso en sí no pueden satisfacer las motivaciones e intenciones personales de cada uno de los alumnos, lo que convierte a la enseñanza en un enfoque uno-para-todos.

No es posible verificar los resultados del aprendizaje

Tercero, no hay manera de **verificar los resultados del aprendizaje**. Puede suceder que los estudiantes aprendan a explotar las propiedades de las plataformas, engañando al sistema y reflejando un aprendizaje que no es real. En el trabajo de Xia et al. [53] se presenta el término ***engañando al sistema*** (del inglés *"gaming the system"*) que es el fenómeno en el que los estudiantes intentan obtener buenos resultados explotando propiedades del sistema en vez de estudiando el material correspondiente. Aunque para evitar esto se suele cambiar todo el flujo de trabajo del curso, generalmente los estudiantes tienden a aprender la nueva forma de engañar al sistema. Un ejemplo de este fenómeno se puede ver en Mumuki, en la Figura 2.3 podemos ver el enunciado del ejercicio a la izquierda, el cual introduce el lenguaje JavaScript a partir de Gobstones, un lenguaje ya conocido por el estudiante al momento de realizar el ejercicio.

En el enunciado se mencionan similitudes y diferencias entre los lenguajes, se dan ejemplos de éstas y por último le pide al estudiante que escriba una función que dado un número, devuelva su mitad. Del lado derecho de la Figura 2.3 podemos ver dos soluciones que conforman un engaño para el sistema, logrando completar el ejercicio sin necesidad de resolverlo correctamente. Del lado superior derecho tenemos la Solución A, donde escribimos una función que siempre devuelve 0. Esta función es testeada por el sistema y falla en cada uno de los casos de test, a lo que Mumuki muestra como feedback en rojo la leyenda "Tu solución no pasó las pruebas" y a continuación muestra todos los casos de test que se le aplicaron a la función. Se puede ver que Mumuki verifica que la función dado un 2 devuelva 1, dado un 20 devuelva 10 y dado un 10 devuelva 5. Debajo de cada uno de estos casos se ve que la verificación se hace comparando el número devuelto por la función y el numero esperado, verificando entonces que 0==1 para el primer caso, 0==10 para el segundo y 0==5 para el tercero, por lo que falla en los tres casos de test. A partir de estos casos de test, podemos engañar a Mumuki y escribir la Solución B que se encuentra en el lado inferior derecho de la Figura 2.3. La función de la Solución B se basa en el input de cada caso de test dado en el feedback de la Solución A y determina para cada uno de ellos el resultado que debe devolver. Por lo tanto, escribimos a mano (en inglés se conoce por el término hard-code) que si el input es un 2 devolver 1, si es un 20 devolver 10 y si es un 10 devolver 5. De esta manera se logra que al enviar la solución Mumuki la considere correcta ya que para cada caso de test la función devolvió el resultado esperado. Esto se puede ver ya en la Solución B el feedback dado por Mumuki es un mensaje en verde con la leyenda "¡Muy bien! Tu solución paso todas las pruebas". Esto es un engaño ya que en realidad no resolvimos el ejercicio de la manera en la

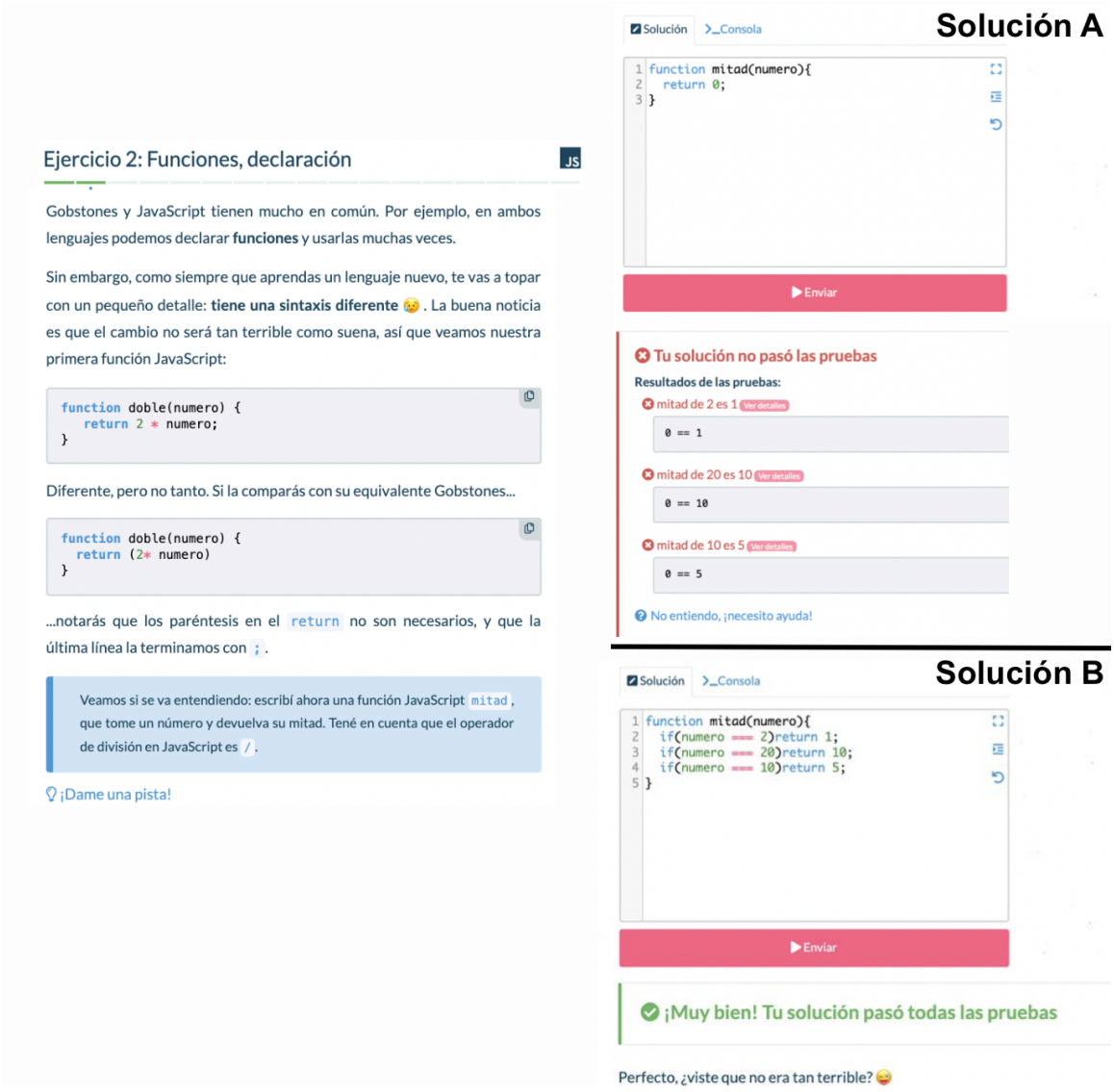


Figura 2.3: Ejemplo de engaño al sistema Mumuki

que el sistema pretendía, nos aprovechamos de que Mumuki nos muestra los casos de test ante un error y resolvimos respondiendo correctamente para cada uno de ellos en lugar de escribir una función que divida el input por 2.

En la sección a continuación detallamos el trabajo previo existente en aprendizaje automático aplicado a conversaciones dadas en entornos de MOOC con el fin de clasificar actos de habla.

2.3.2. Trabajo previo en clasificación automática de actos de habla en foros educativos

Es de interés para este trabajo entender trabajos previos realizados que usan aprendizaje automático para asistir la tarea docente en el marco de conversaciones educativas entre estudiantes y mentores como los realizados por Rus et al. [40], Arguello y Shaffer [2] y Kim y Kang [23]. Describimos los mismos para poder analizar resultados obtenidos, técnicas utilizadas, características consideradas y experimentos realizados previamente en la clasificación automática de actos de habla. También nos es útil para considerar distintos puntos de vista y enfoques de otros investigadores que ya hayan abordado el problema anteriormente.

En el trabajo de Vasile Rus et al. [40] se busca lograr la detección de clases de actos de habla utilizando algoritmos de *clustering*. El clustering es la clasificación no supervisada de datos (comúnmente representados como vectores en un espacio multidimensional) dentro de grupos (*clusters*) basado en sus similitudes. Por lo tanto, un cluster es una colección de objetos que son similares entre los que pertenecen al mismo cluster pero disímiles a los objetos pertenecientes a otros clusters. Los datos sobre los que se trabaja son conversaciones provenientes de tres juegos educativos llamados *Urban Science*, *Land Science* y *Nephrotex*, obteniendo un conjunto de datos de 2768, 4131 y 1000 mensajes respectivamente, clasificados manualmente por expertos en 8 clases de actos de habla. En estos juegos los estudiantes simulan distintos tipos de escenarios donde aprenden sobre ecología y urbanidad, y tienen a su disposición un chat con mentores donde conversan acerca de las reglas del juego, el contenido, dudas, sugerencias, etc. Rus et al. sugieren utilizar como input para los algoritmos de clustering las primeras palabras (comúnmente llamadas *tokens*) de cada declaración, ya que intuitivamente presentan una gran cantidad de información relacionada con el acto del habla presente en la declaración. Esta decisión está principalmente fundada en el hecho de que ciertas categorías de actos de habla siguen ciertos patrones, por ejemplo, los agradecimientos suelen ser un conjunto pequeño de palabras y expresiones que todas las personas utilizan, también las preguntas, que generalmente comienzan con palabras como *"¿Qué-"*, *"¿Cómo-"*, *"¿Cuándo-"*, *"¿Dónde-"*,

etc. Sumado a esto, generalmente en una conversación el oyente comienza su turno inmediatamente después de que termina el turno anterior o en algunos casos antes, lo que sugiere que la mayor información del acto del habla está presente al comienzo de la declaración. En [40] se realizaron experimentos con dos algoritmos de clustering, *Expectation-Maximization (EM)* [44] y *K-Means* [21]. Teniendo en cuenta que la principal diferencia entre ambos es que en K-Means, un hiper parámetro es la cantidad de clusters que se desean obtener, se utilizó primero EM, dónde se obtuvieron los mejores resultados utilizando los tres primeros tokens de cada declaración, con el mejor accuracy en el conjunto de datos del juego *Urban Science* de 40.3 % identificando 6 clusters, mientras que adivinando al azar es del 12.5 %. Con esto, se utilizó K-Means variando la cantidad de clusters en valores cercanos a la clasificación hecha a mano (8 clases), siempre utilizando los primeros tres tokens de cada declaración. A partir de este trabajo podemos notar que la utilización de algoritmos de machine learning es más efectiva a la hora de clasificar actos de habla en una declaración que adivinar aleatoriamente. A diferencia de Rus et al., en este trabajo no utilizaremos algoritmos de clustering, además de que nuestra clasificación final consiste en si el acto de habla es un forward o backward looking act.

En el trabajo de Arguello y Shaffer [2], se analizaron datos provenientes de foros de la Universidad de Carolina del Norte. Obtuvieron un conjunto de datos de 2943 mensajes escritos por estudiantes y mentores. Se tuvieron en cuenta 7 clases de actos de habla: pregunta, respuesta, problema, resolución de problema, reconocimiento positivo (del inglés *positive acknowledgement*), reconocimiento negativo (del inglés *negative acknowledgement*) y otro. En este trabajo se realizan tres análisis de regresión logística [33] previo a la clasificación de los actos de habla, que exploran las siguientes preguntas: ¿Los actos de habla ayudan a identificar discusiones que requieren la atención de un mentor?, ¿Ayudan a predecir si un estudiante va a completar un ejercicio?, ¿Ayudan a predecir la nota de un estudiante en un ejercicio? En cada una de las tres preguntas se utilizaron pruebas de Wald[11] para encontrar una respuesta a partir de características presentes en las discusiones y de datos relacionados al comportamiento en el foro de los autores de los mensajes analizados. Para responder la primera pregunta, la prueba de Wald obtuvo como resultado que un mensaje es 2.278 veces más probable que sea escrito por un mentor cuando el mensaje anterior era un acto de habla clasificado como problema y 3.053 veces más probable cuando el mensaje anterior era un acto del habla clasificado como negative acknowledgement. También, un mensaje es 1.406 veces más probable que sea escrito por un mentor por cada mensaje previo en la discusión que haya sido clasificado como un problema.

[11] https://es.wikipedia.org/wiki/Prueba_de_Wald

Otro resultado es que un mensaje es 1.562 veces más probable que sea escrito por un mentor por cada mensaje previo escrito por algún mentor en la discusión. Para la segunda y tercera pregunta los modelos planteados explican muy poco de la varianza, por lo que se concluye que los actos de habla no fueron lo suficientemente útiles para predecir la resolución de un ejercicio y el rendimiento en el mismo. Esto se puede deber a que los estudiantes de los MOOC tienen distintas bases y se inscriben por diversos motivos [10], lo cual no se ve reflejado en este tipo de conjunto de datos. Un modelo que predice el rendimiento en los MOOC basado en los actos de habla presentes en sus foros podría necesitar considerar el nivel de conocimiento previo de los estudiantes a la hora de inscribirse. La predicción de que un mentor responda en una consulta es una tarea más simple utilizando los actos de habla, ya que éste interviene basándose en lo que ve en el foro y por lo tanto el conjunto de datos de los mensajes tiene información más relevante para esta tarea. Luego de haber respondido las tres preguntas planteadas, Arguello y Shaffer [2] implementan la clasificación de actos de habla aplicando regresión logística con regularización L2 para clasificar los mensajes en 7 clases de actos de habla. Teniendo en cuenta diferentes características y haciendo combinaciones de ellas, se obtuvo en el peor de los casos: 8.2 % de accuracy para el acto de habla de negative acknowledgement, aunque luego los accuracy varían entre el 49 % para la resolución de problemas y el 75 % para positive acknowledgement. [2] nos sirve para ver que se pueden obtener buenos resultados en clasificación del habla con una regresión logística, aunque este trabajo se diferencia del nuestro en el método a utilizar y en la clasificación final. Este trabajo a partir de la primera pregunta planteada nos es útil para poder considerar que la utilización de actos de habla en mensajes es efectivo como herramienta para clasificar si un mensaje va a necesitar la atención de un mentor o no.

En el trabajo de Kim y Kang [23] se analizan 1135 mensajes en foros de una materia de Sistemas Operativos de la Universidad de California del Sur. Kim y Kang hacen hincapié en que este tipo de conjunto de datos tiene un cierto nivel de complejidad ya que las discusiones de estudiantes son muy informales y ruidosas con respecto a gramática, sintaxis y puntuación. Hay mucha varianza en la forma en la que los estudiantes presentan información similar, cuando se habla de ejercicios de programación se incluyen varias referencias a código de programación, e incluso algunos datos son mensajes de humor o anuncios personales. En este sentido los datos de este trabajo son similares a los nuestros. En este trabajo se tiene una mirada de clasificación de actos de habla como pares de adyacencia (concepto que se introdujo en la Sección 2.1.2), donde un acto del habla define el rol que un mensaje juega contra otro mensaje al cual responde. Un mensaje puede jugar diferentes roles al

mismo tiempo, puede incluir una pregunta para un problema en particular, o puede contener una respuesta o sugerencia con respecto a una pregunta previa en la conversación. En total se definen 5 clases de actos de habla: preguntas, respuestas con sugerencias o recomendaciones, problemas o desentendimientos, positive acknowledgement y negative acknowledgement. Luego, se plantean 6 características a tener en cuenta para el entrenamiento del modelo: frases claves en los mensajes que infieren el acto del habla presente y su posición tanto para el mensaje actual como para el anterior, la posición del autor del mensaje actual y el anterior en el orden en el que los autores fueron llegando a la conversación, posición del mensaje actual y anterior en la conversación. Con esto, Kim y Kang crearon clasificadores binarios para cada acto del habla, donde cada clasificador chequea que cierto acto del habla exista para un par de mensajes. Los algoritmos utilizados fueron SVM [9], J48 [43] y Naive Bayes [50], obteniendo F1 scores en SVM de 88.6 % para las preguntas, 85.0 % para las respuestas, 58.8 % para los problemas o desentendimientos y 50.0 % para los positive acknowledgement, no hay datos para los negative acknowledgement dado que representan menos del 3 % de los datos. Finalmente, Kim y Kang buscan poder determinar si una conversación quedó con preguntas sin contestar o con problemas sin resolver, es decir, detectar si se necesita la ayuda de un mentor. Para esto, realizan un análisis en la estructura de las conversaciones, identificando patrones. Ejemplos de esto pueden ser sucesiones pregunta-respuesta, donde se responde directamente la pregunta, o pregunta-pregunta-respuesta-respuesta donde la segunda pregunta se hace para solicitar más detalles, la primera respuesta responde a la segunda pregunta y la segunda respuesta responde a la pregunta inicial o pregunta-respuesta-positive acknowledgement, etc. Con este reconocimiento de conversaciones como estructuras, se plantean 4 preguntas a resolver con la clasificación de los actos de habla. La primera: ¿Todas las preguntas fueron respondidas? Esta pregunta se responde detectando si hubo algún acto del habla "pregunta" en la discusión y para cada una de ellas ver si hubo un acto del habla "respuesta" y que la estructura de la discusión (visto como una cadena o árbol) no termine con una pregunta sin una respuesta. La segunda: ¿Hubo algún problema o confusión? Esta pregunta se responde detectando el acto del habla "problemas o desentendimiento" en la discusión. La tercera: ¿Fueron esos problemas o confusiones resueltos? Esta pregunta se responde detectando los problemas o desentendimientos en la discusión y revisando que para cada una de ellas haya una respuesta y que la estructura no termine en un problema o desentendimiento. La cuarta: ¿La primera respuesta satisfizo la primera pregunta? Esta pregunta se responde detectando si hubo una respuesta para la primera pregunta y si el autor de la primera pregunta no tuvo preguntas, problemas o inconvenientes después de

la respuesta. Luego de tener planteadas estas 4 preguntas, se determinan variadas características de la estructura de los actos de habla de la conversación, por ejemplo, si hubo alguna pregunta en la discusión, si hubo alguna respuesta al primer mensaje, si dado un autor éste escribió un mensaje con un positive acknowledgement, etc. Con todas estas características, crearon un clasificador para responder a cada una de las 4 preguntas planteadas por cada discusión, logrando un accuracy del 89.3 % para la primera pregunta, 38.4 % para la tercera pregunta y 95.5 % para la cuarta pregunta. La segunda pregunta no se tuvo en cuenta ya que requiere de que la conversación solamente tenga problemas o desentendimientos. La tercera pregunta tuvo bajo accuracy debido al problema mencionado previamente con respecto al clasificador de problemas o desentendimientos. De esta manera, el trabajo de Kim y Kang nos es útil para ver que se puede obtener buenos resultados en la clasificación automática de actos de habla y se diferencia del nuestro en el método de desarrollar una heurística sobre la estructura de las conversaciones para determinar si se necesita la atención de un mentor.

2.4. Relación con otros capítulos

En este capítulo analizamos las propiedades y la estructura de la conversación humana, vimos que en las declaraciones que se dan en una conversación existen actos de habla, los cuales definimos y analizamos sus tipos. A partir de la existencia de los actos de habla vimos que podemos distinguir las declaraciones en forward y backward looking acts. Luego analizamos las conversaciones en un contexto educativo, presentamos el concepto de MOOC, vimos sus beneficios, puntos críticos e introdujimos al MOOC que será la base de este trabajo: Mumuki. Por último analizamos tres trabajos previos en la tarea de clasificación automática de actos de habla en foros educativos. En el Capítulo 3 analizaremos con mayor nivel de detalle al MOOC Mumuki introducido en este capítulo, su funcionamiento y detallaremos el conjunto de datos proveniente de los foros educativos de Mumuki, los cuales se utilizaran en este trabajo.

Capítulo 3

Mumuki

En este capítulo se describe el funcionamiento del sistema online Mumuki para aprender a programar, el funcionamiento de su foro de consultas y se realiza un análisis previo del conjuntos de datos.

En la Sección 3.1 se detalla el funcionamiento del sistema online Mumuki, qué tipos de usuarios forman parte de la plataforma, cómo se accede al foro de consultas, qué es una consulta y cómo realizarla. En la Sección 3.2 se especifican los posibles estados que puede tener una consulta y de qué manera se puede cambiar el estado de ésta. También en la Sección 3.3 se describen las heurísticas que Mumuki utiliza sin soporte de aprendizaje automático para poder detectar mensajes que necesiten atención de mentores. Por último, en la Sección 3.4 se describe el conjunto de datos disponible y sobre el que trabajaremos en esta tesis.

3.1. Mumuki: Funcionamiento del sistema y de su foro de consultas

En esta sección se detalla el sistema online Mumuki, el funcionamiento general de la plataforma y del foro de consultas, espacio donde los estudiantes se conectan con mentores y otros estudiantes con el fin de poder discutir sobre los ejercicios a resolver. Para este trabajo es importante el trabajo realizado por Benotti et al. [5], donde se compara el aprendizaje del lenguaje de programación Haskell entre cursos que utilizan Mumuki y cursos donde no lo utilizan. En particular, en [5] también se detalla el funcionamiento de Mumuki y la retroalimentación (en inglés: *feedback*) automática que este devuelve.

Mumuki[1] es un sistema online utilizado para la enseñanza de programación. Su contenido educativo consiste de un conjunto de capítulos que abarcan distintos co-

[1] https://mumuki.io/

nocimientos previamente considerados por los docentes que son apropiados para el aprendizaje del estudiante. En la Figura 3.1 se puede ver cómo un estudiante ve la organización del contenido en distintos capítulos. El estudiante debería resolver idealmente en el orden dispuesto por el sistema, pero no es obligatorio.

Figura 3.1: Capítulos disponibles para aprender en Mumuki de forma autodidacta

El orden dispuesto por el sistema está dado por la numeración de los capítulos, en forma creciente, cada uno con su respectivo título representativo de qué contenidos abarca, acompañado de una imagen que da una intuición acerca del tema tratado en el capítulo y un breve texto introductorio. Cuando un estudiante accede a uno de los capítulos, se encuentra con una lista de ejercicios a resolver, organizados en distintas lecciones, cada uno con su respectivo título. Los ejercicios también tienen un orden dispuesto por el sistema que está dado por la numeración de los mismos, en forma creciente, por cada una de las lecciones del capítulo. En la Figura 3.2 podemos observar cómo un estudiante ve la organización de los ejercicios del capítulo.

Los ejercicios se presentan organizados en columnas, con una imagen de un círculo a su izquierda y su título es una referencia al contenido a aprender en el ejercicio. La idea de que el estudiante resuelva los ejercicios en el orden establecido también se da por el hecho de que éstos no son independientes unos de otros, sino que muchas veces se relacionan entre sí a modo de ir introduciendo conceptos cada vez más complejos, basándose en los conceptos de un ejercicio anterior.

Cuando el estudiante elige uno de los ejercicios disponibles, Mumuki presenta el ejercicio como se puede observar en la Figura 3.3. En la parte superior muestra el número de ejercicio y su título. Debajo de esto nos muestra una barra de progreso que consiste de tantos rectángulos como ejercicios tenga la lección, en orden creciente

Figura 3.2: Organización de ejercicios en uno de los capítulos

de izquierda a derecha. Los ejercicios ya resueltos se muestran con un rectángulo de color verde, los que aún no fueron realizados se muestran con un rectangulo de color gris. El ejercicio que el estudiante está realizando se marca con un círculo azul debajo del rectángulo correspondiente al ejercicio. Luego, en la columna de la izquierda se muestra la descripción del ejercicio, y en la columna de la derecha se muestra un editor de texto donde el estudiante escribe el programa que resuelve el ejercicio, y abajo del editor de texto se cuenta con un botón rojo con texto "Enviar" con el cual el estudiante solicita la revisión del programa escrito. En el caso de la Figura 3.3, el programa que resuelve el ejercicio es el que se ve en el editor de texto. En este ejercicio en particular, el programa está escrito en un lenguaje de programación llamado Gobstones[2], un lenguaje pensado y diseñado para la enseñanza introductoria a la programación. Ante este caso de resolución correcta, luego de hacer click en el botón "Enviar", Mumuki muestra un mensaje en verde que dice que la solución enviada pasó todos los casos de prueba que se le aplican al programa enviado para probar su correctitud, junto con una imagen que facilita la comprensión del resultado obtenido. Este resultado es la retroalimentación que Mumuki da al estudiante de la resolución del ejercicio, y a partir de ahora lo llamaremos por su nombre en inglés *feedback*. Al código, feedback y toda la información relacionada con la evaluación hecha por Mumuki sobre un ejercicio enviado por el estudiante para verificar su correctitud lo llamaremos *submission*.

Cuando un ejercicio no está bien resuelto es cuando Mumuki permite el acceso a

[2]http://gobstones.github.io

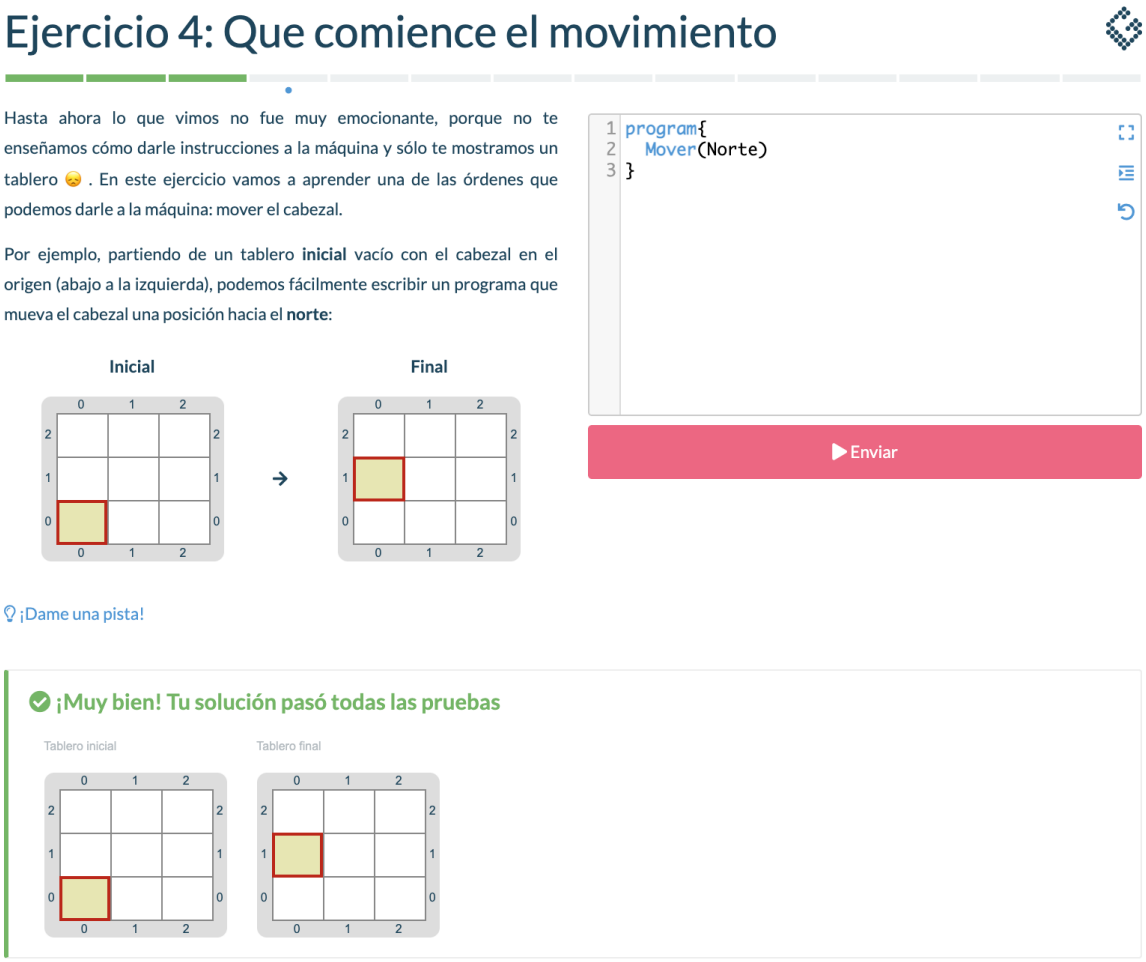


Figura 3.3: Ejercicio en Mumuki con una resolución correcta

su foro de consultas. En la Figura 3.4 vemos que si cambia el código que era correcto, por uno que no lo es, y se hace click en "Enviar", en la barra superior de rectángulos correspondientes a los ejercicios, se muestra en rojo el ejercicio actual, lo que significa que el ejercicio no fue resuelto correctamente, además, Mumuki devuelve el feedback de error debajo de las columnas de descripción y de editor de texto en color rojo con la leyenda "Tu solución no pasó las pruebas", con una descripción breve del motivo por el cual el programa no es correcto, seguido de tres imágenes que hacen referencia a la situación inicial, la situación final a la que se esperaba llegar con el programa, y la situación final obtenida. El estudiante puede continuar con sus intentos de resolver el ejercicio mediante la edición del programa escrito en el editor de texto, y apretando en el botón "Enviar" la cantidad de veces que necesite. También se puede optar por hacer click en "¡Dame una pista!", ante lo cual Mumuki extenderá levemente la descripción del ejercicio con más información para orientar al estudiante.

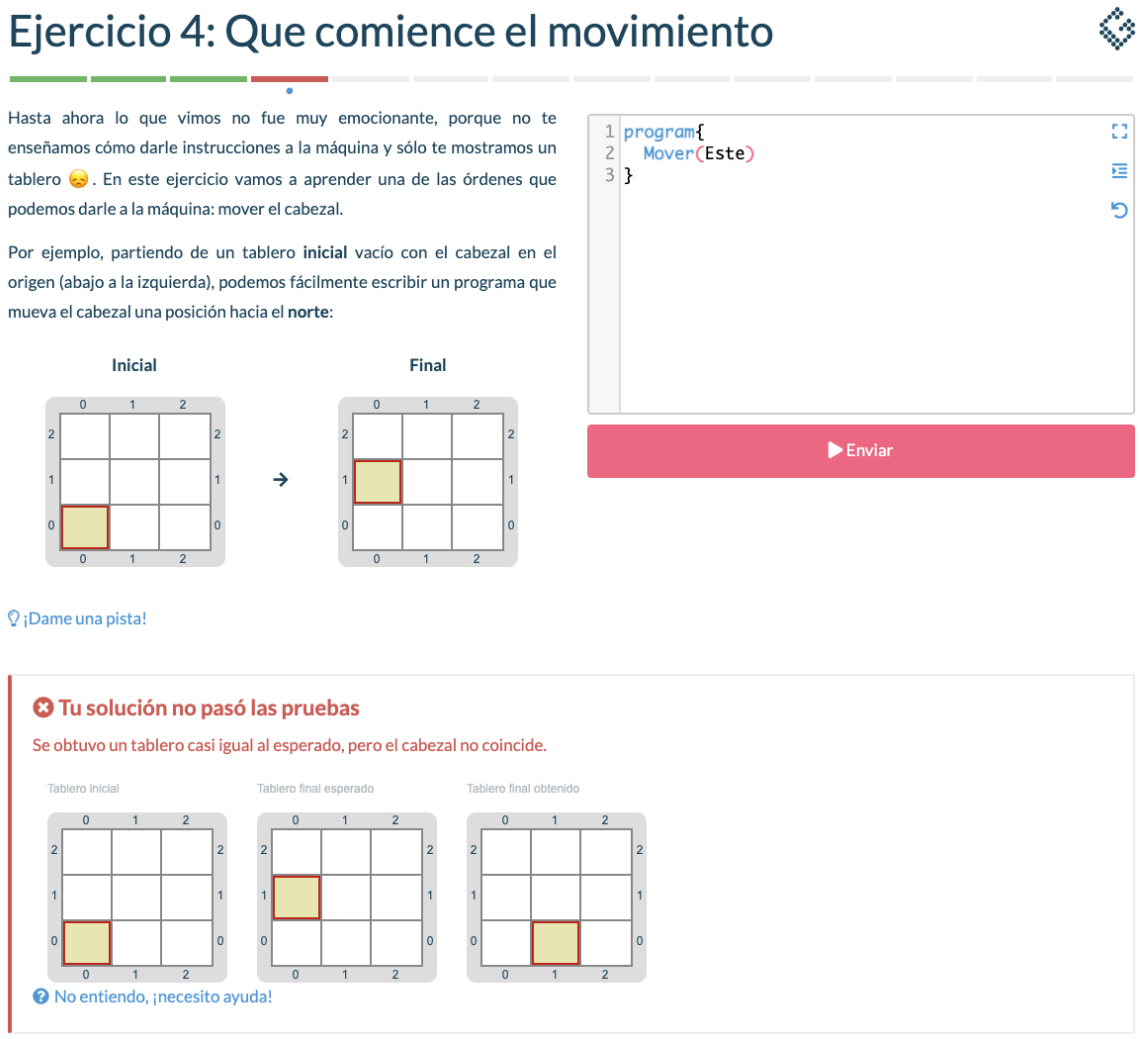


Figura 3.4: Ejercicio en Mumuki con una resolución incorrecta

En caso de que el estudiante no pueda resolver el ejercicio por su propia cuenta, puede optar por la opción de acceder al foro de consultas haciendo click en "No entiendo, ¡Necesito ayuda!". De esta manera, una consulta siempre está relacionada con un ejercicio. Dentro de los usuarios que utilizan el foro, podemos distinguir dos tipos: mentor o estudiante. Un estudiante es una persona que está cursando los contenidos del sistema, con el fin de aprenderlos. Un mentor es un docente capacitado para dar una respuesta correcta a cualquier consulta que un estudiante realice. A su vez tiene más permisos dentro del foro (por ejemplo, puede borrar mensajes que violen las condiciones de uso).

Antes de poder realizar una consulta, Mumuki muestra al estudiante todas las consultas que fueron realizadas con respecto al ejercicio que el usuario está intentando realizar, con el fin de que el estudiante revise las dudas que a otros estudiantes les

surgieron y que plasmaron en consultas, y qué respuestas recibieron de los demás estudiantes y mentores. Si esto no es de utilidad, el estudiante puede proceder a crear su consulta. Al crear su consulta, el estudiante deja escrito un mensaje explicando su duda, y queda adjunta a la consulta la *Solución*: programa escrito por el estudiante sobre el cual se tiene una duda, el *Resultado*: feedback que Mumuki dió al programa enviado y la *Descripción*: enunciado del ejercicio.

Luego de que se realiza la consulta, esta queda pública para todos los estudiantes y mentores que accedan al foro para el mismo ejercicio. Una consulta puede ser respondida por estudiantes y mentores, formándose así una **conversación** para poder ayudar al estudiante creador de la consulta a resolver correctamente el ejercicio. A cada una de las declaraciones realizadas en una consulta del foro le llamaremos mensaje. Cada mensaje está relacionado con una consulta. En la Figura 3.5 podemos ver una consulta realizada por un estudiante llamado Matías acerca del ejercicio 4.

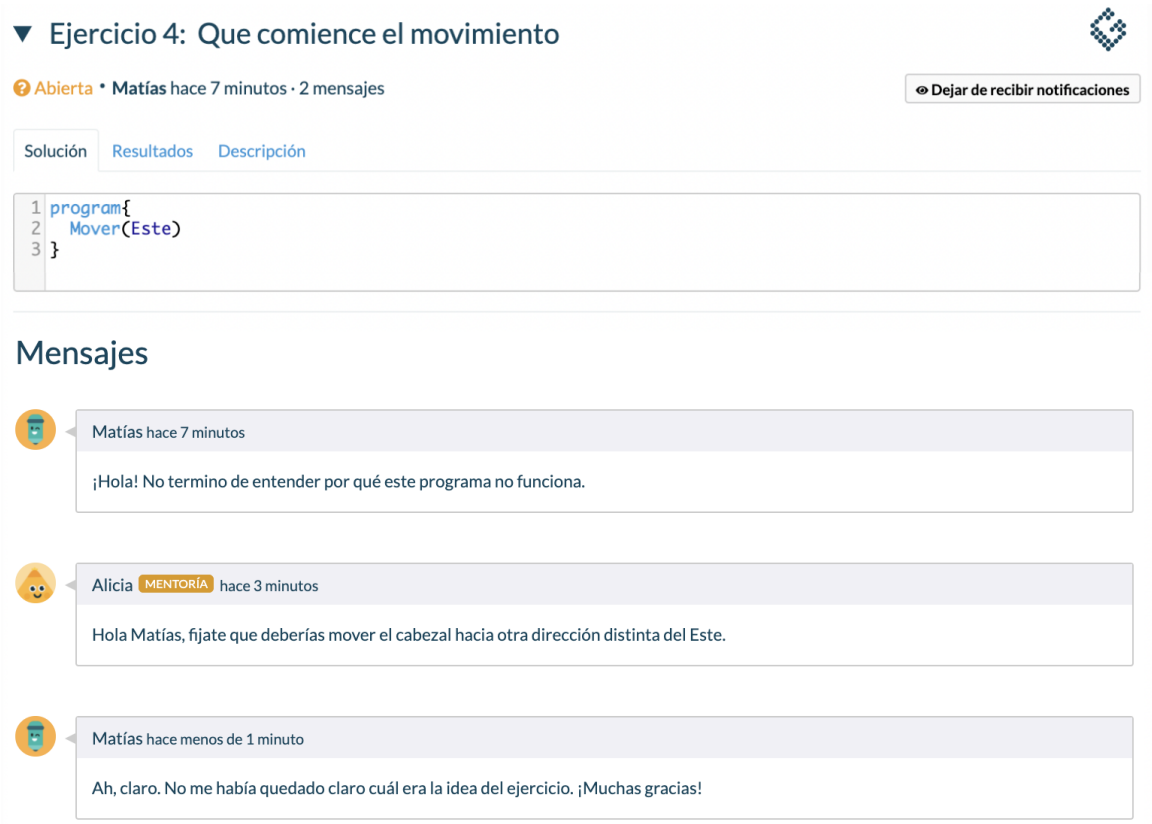


Figura 3.5: Consulta en el foro: conversación entre un estudiante y un mentor

En la consulta de Matías de la Figura 3.5 podemos ver que se muestra la conversación que se desarrolló en la consulta, como primer mensaje figura la duda del creador de la consulta, en este caso, Matías. Luego, una mentora de Mumuki llamada

Alicia sugirió una pista para que Matías comprenda mejor el ejercicio. Por último, Matías responde agradeciendo la pista a Alicia y finaliza la conversación. En el caso del primer mensaje de Matías, se trata de un mensaje que es un forward looking act, ya que expresa desentendimiento y genera una obligación en el oyente de atender esa duda. La atención de esta obligación generada por el mensaje de Matías es llevada a cabo por Alicia, la escritora del segundo mensaje de la conversación, quien contesta la duda de Matías dando una pista del ejercicio. El mensaje de Alicia es un backward looking act: no genera una obligación en el oyente de respuesta o atención, la intención de Alicia de ayudar a Matías empieza y termina en el mensaje mismo, sin intención de buscar atención por parte de Matías o alguna otra persona. Por último, Matías responde dando a entender que la pista de Alicia lo ayudó a comprender el ejercicio y agradeciéndole por su respuesta. Este mensaje de Matías es un backward looking act, al igual que el mensaje de Alicia: no genera obligación de atención por parte del oyente. Un mentor tiene la capacidad de marcar las respuestas de otros estudiantes como válidas, con el fin de poder indicarle al estudiante que realizó la consulta que la respuesta de otro estudiante es correcta y así evitar repetir respuestas. El estudiante que creó la consulta, los demás estudiantes y los mentores pueden escribir en la conversación de la consulta la cantidad de veces que haga falta, sin límite.

En la sección a continuación se detallan los posibles estados que una consulta puede tener y las transiciones entre ellos.

3.2. Estados de las consultas

En esta sección se detallan los distintos estados que las consultas pueden tener, como también las formas de transicionar entre ellos.

Dentro del foro de consultas, se pueden distinguir cuatro posibles estados para una consulta.

- **Abierta**: Estado por defecto al momento de crear la consulta, donde se da lugar a una conversación donde puede participar cualquier usuario.
- **Cerrada**: Estado de aquellas consultas cuyo ejercicio fue resuelto con éxito por el mismo creador, posteriormente a la creación de la consulta, sin que haya habido participación de otro usuario cualquiera en la conversación. No son públicamente visibles.
- **Resuelta**: Estado de aquellas consultas en las que el creador pudo resolver el ejercicio posteriormente. Son públicamente visibles con el fin de servir de guía para los demás alumnos que tengan dudas similares.

- **En revisión**: Estado de aquellas consultas cuyo ejercicio fue resuelto con éxito por el mismo creador, posteriormente a la creación de la consulta, cuando haya habido participación de otro usuario cualquiera en la conversación. De este estado se pasa al estado de "cerrada" o "resuelta" según la aprobación de un mentor. No son públicamente visibles.

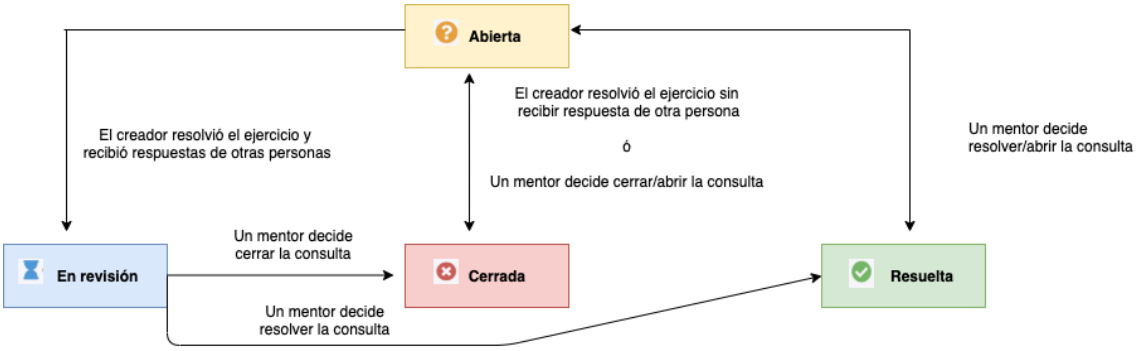


Figura 3.6: Transiciones de estados de consulta

Un mentor puede resolver o cerrar una consulta que está abierta, y a su vez puede reabrir una consulta que está resuelta o cerrada. Esto implica que un mentor tiene la facultad de poder cambiar el estado de las consultas por decisión propia. En la Figura 3.6 podemos ver un diagrama de transiciones de estado de las consultas. Por defecto, al crear una consulta su estado es "Abierta". Luego, si el creador de la consulta resolvió el ejercicio posteriormente, sin recibir respuesta de otro estudiante o un mentor, se cierra la consulta, entonces la consulta pasa a estado "Cerrada", otro posible caso para pasar una consulta de "Abierta" a "Cerrada" es que un mentor decida cerrarla. A su vez, un mentor puede pasar una consulta de "Cerrada" a "Abierta" abriendo la consulta. Para pasar una consulta de "Abierta" a "Resuelta" la única forma es mediante un mentor que decida resolver la consulta, y viceversa, para pasar una consulta de "Resuelta" a "Abierta", un mentor decide abrir la consulta. Si estando la consulta abierta, el estudiante resuelve el ejercicio pero algún mentor o estudiante contestó a la consulta antes, la consulta pasa al estado "En revisión". A partir de que una consulta entra en estado de revisión, un mentor debe decidir si cerrar la consulta, por lo que pasa de estado "En revisión" a "Cerrada", o resolverla, por lo que pasa de estado "En revisión" a "Resuelta".

En la sección siguiente se detallan las herramientas que Mumuki posee, al momento de realizar este trabajo, para intentar detectar mensajes de estudiantes que necesitan atención.

3.3. Detección de mensajes que necesitan atención en Mumuki

En esta sección se describen los métodos que Mumuki implementa para intentar filtrar aquellos mensajes que requieren la atención de un mentor. Ninguno de estos métodos incluyen utilización de aprendizaje automático. El objetivo del filtrado es facilitar a los mentores el trabajo de identificar aquellos mensajes de estudiantes que necesitan ayuda para poder resolver los ejercicios.

A la hora de visualizar los mensajes de las consultas, el mentor tiene acceso a opciones que desde la interfaz del estudiante no están disponibles. Las opciones son las siguientes.

- **Marcar mensaje como respuesta válida**: Sirve para que el mentor pueda indicar que una respuesta dada por otro estudiante es una respuesta correcta, ya sea porque el mentor coincide con ella (y hubiera respondido lo mismo) y/o la considera acertada.
- **Marcar mensaje como "no es una pregunta"**: Sirve para marcar que una pregunta no requiere atención. Por ejemplo: *"Muchas gracias, me sirvió tu respuesta"* o *"Ya entendí!"*.
- **Eliminar mensaje**

En la Figura 3.7 podemos ver la misma consulta que en la Figura 3.5, sólo que desde el punto de vista de un mentor. En el encabezado de cada mensaje, sobre el lado izquierdo se puede ver el nombre de la persona que envió el mensaje y el tiempo transcurrido desde su envío, luego sobre el lado derecho se pueden ver las tres opciones antes mencionadas. La opción "Marcar mensaje como respuesta válida" es el primer ícono con forma de tick, la opción "Marcar mensaje como "no es una pregunta"" es el segundo ícono con forma de signo de interrogación tachado, y la opción "Eliminar mensaje" es el tercer ícono con forma de tacho de basura.

En Mumuki, la opción de marcar un mensaje como "No es una pregunta" es utilizada dentro de una heurística interna para poder determinar qué mensajes requieren la atención de un mentor, para poder darles prioridad a estos mensajes antes que a otros. Por defecto, cada vez que un mensaje es enviado, es considerado "una pregunta", es decir, la opción "No es una pregunta" está desmarcada por defecto cuando un mensaje es creado. La intención de Mumuki es utilizar este campo para determinar que, si el último mensaje enviado en una consulta es de un estudiante, y el mensaje está marcado como una pregunta (que es la opción por defecto al crear el mensaje), entonces la consulta tiene un mensaje que requiere atención. En la práctica esta

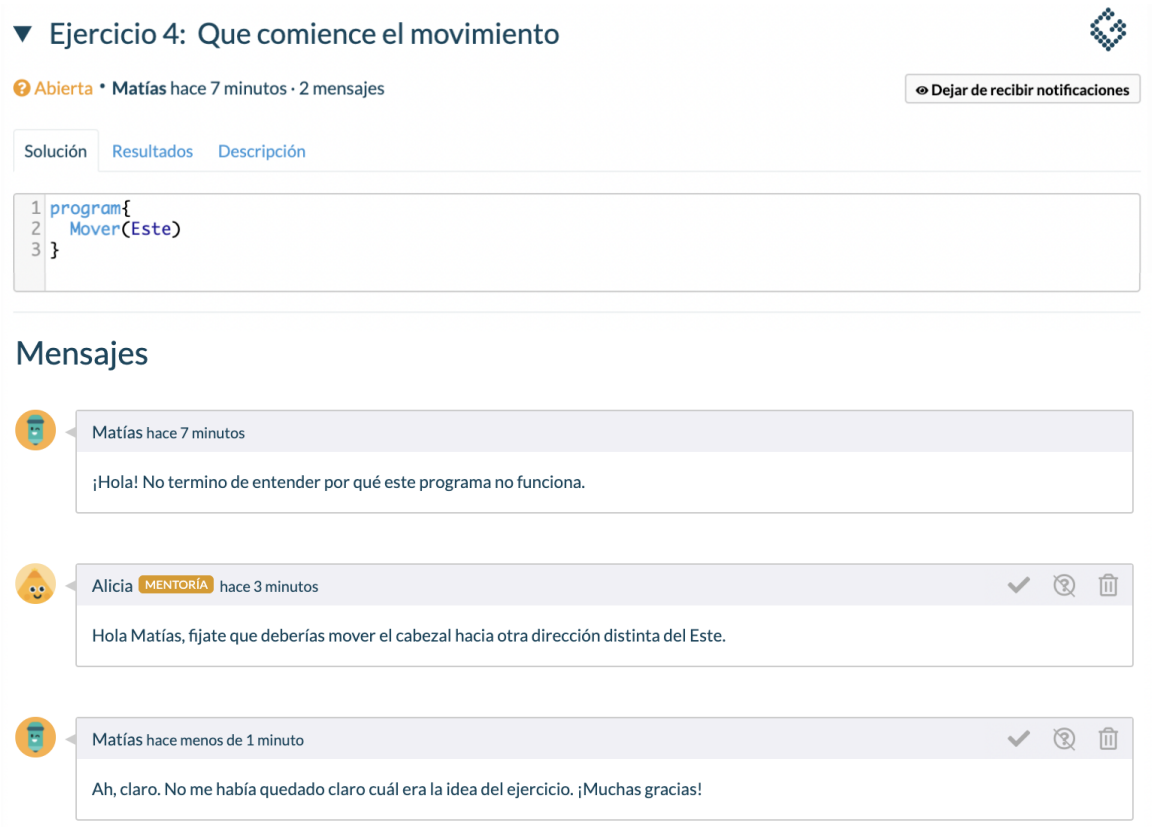


Figura 3.7: Mensajes en una consulta desde el punto de vista de un mentor

heurística no fue utilizada dado que al momento de analizar el conjunto de datos la opción de "Marcar un mensaje como "no es una pregunta"" tenía poco tiempo de existencia en la plataforma.

Además de la heurística recién mencionada, Mumuki cuenta con una opción de filtrado de consultas de acuerdo a si contienen mensajes que requieren atención. Esta opción de filtrado está disponible en el menú donde se listan todas las consultas que existen con respecto a un ejercicio en particular.

En la Figura 3.8 se puede ver el menú de consultas de Mumuki con respecto a un ejercicio en particular, en este caso para el "Ejercicio 4: Que comience el movimiento". En el lado derecho de la barra gris superior, Mumuki permite utilizar el filtrado de consultas "Requiere atención" que funciona bajo la siguiente heurística: la consulta contiene un mensaje que requiere atención si el último mensaje enviado no es un mensaje de un mentor ni un mensaje marcado como respuesta válida.

De este modo, Mumuki intenta, a través de la heurística del marcador "no es una pregunta" y del filtrado "No requiere atención", disminuir el tiempo que los mentores pasan buscando consultas con mensajes a los cuales atender. Sin embargo,

Figura 3.8: Espacio de consultas de un ejercicio

estos métodos no son efectivos, ya que en el caso del marcador "no es una pregunta", se asume que cualquier mensaje enviado por un estudiante es una pregunta a priori, lo cual no siempre es cierto. El filtrado "No requiere atención" no es utilizado en la realidad por los mentores, simplemente marcan la consulta como cerrada cuando detectan que contiene mensajes que no requieren atención.

En la sección a continuación se analiza el conjunto de datos disponible.

3.4. Conjunto de datos

En esta sección analizaremos el conjunto de datos disponible que Mumuki nos provee, junto con un análisis estadístico del mismo.

Contamos con los datos correspondientes a: consultas realizadas, mensajes enviados en las consultas, usuarios y ejercicios. Debido a la extensión de los campos de cada conjunto, en esta sección se detallan los campos que son relevantes para nuestro trabajo, dejando la explicación extensiva de las consultas en el Apéndice A y de los mensajes en el Apéndice B.

Para nuestro trabajo es relevante el análisis de los siguientes campos. En las consultas:

- **id**: Identificador de la consulta.
- **description**: Texto del mensaje que escribió la persona cuando creó la consulta (a partir de ahora también mencionado como mensaje inicial).
- **initiator_id**: Identificador de la persona que creó la consulta.

Luego, en los mensajes son de interés los siguientes campos:

- **id**: Identificador del mensaje.
- **discussion_id**: Identificador de la consulta donde se envió el mensaje.
- **content**: Texto del mensaje.
- **sender_uid**: Identificador de la persona que escribió el mensaje.

Por último, de los datos de usuarios nos interesa:

- **uid**: Identificador del usuario.
- **created_at**: Fecha de registro del usuario en Mumuki (permite discernir entre un estudiante y un mentor).

A partir del conjunto de datos y teniendo en cuenta los campos mencionados, se pueden reconstruir las conversaciones y realizar análisis sobre las mismas. En la Tabla 3.1 tenemos la cantidad total de elementos en cada tabla provista por Mumuki.

Cantidad de consultas	4242
Cantidad de mensajes iniciales	4242
Cantidad de mensajes no iniciales	5324
Cantidad de ejercicios	151
Cantidad de usuarios	1262

Tabla 3.1: Cantidad de datos en cada tabla

Todas las consultas fueron creadas en el contexto de un plan de 151 ejercicios creado por Mumuki disponibles para la resolución de los estudiantes, en este plan participaron 1262 personas: 8 mentores y 1254 estudiantes.

Se tiene un total de 4242 consultas creadas, donde hubo al menos una consulta realizada para cada uno de los 151 ejercicios. Para cada consulta siempre existe un *mensaje inicial* que el creador de la consulta escribe explicando su duda, y luego pueden existir más mensajes *(no iniciales)* propios de la conversación, como pueden ser respuestas de otros estudiantes, respuestas de mentores, mensajes de otros estudiantes sumándose a la duda, incluso mensajes del estudiante creador de la consulta pidiendo más información o intentando aclarar respuestas recibidas, entre otros. Por ejemplo, en la Figura 3.5, el primer mensaje de Matías es el mensaje inicial, mientras que los demás mensajes posteriores son los mensajes no iniciales. En nuestro conjunto de datos se enviaron un total de 5324 mensajes no iniciales. En total, los mensajes iniciales y no iniciales suman un total de 9566 mensajes.

A partir de la tabla de consultas y de mensajes, podemos analizar la longitud en caracteres de los mensajes iniciales de las consultas y la longitud en caracteres de los mensajes que no son mensajes iniciales. Asumimos que el mensaje inicial de una consulta siempre es un forward looking act: no pueden ser backward looking acts porque no hay mensajes anteriores a los que "mirar hacia atrás".

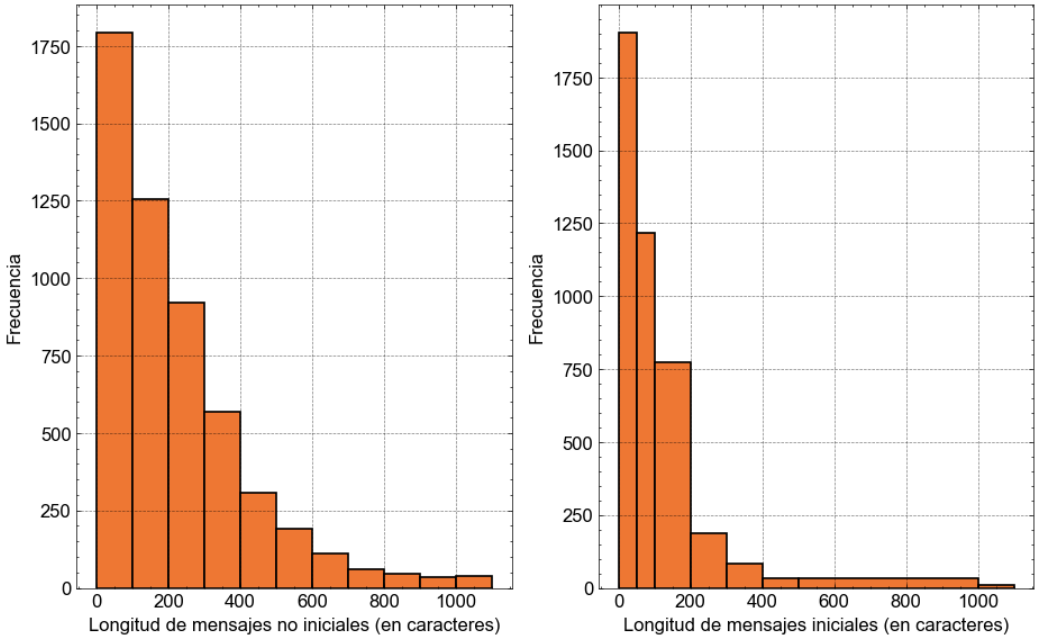


Figura 3.9: Histograma: longitud de mensajes en el foro de consultas de Mumuki

En la Figura 3.9 se puede ver del lado izquierdo el histograma correspondiente a la longitud en caracteres de mensajes no iniciales: hay 3047 mensajes con menos de 200 caracteres de los 5324 que se enviaron, esto es el 57.2 % de los mensajes. Mientras tanto, del lado derecho se puede ver el histograma correspondiente a la longitud en caracteres de los mensajes iniciales. Hay 3895 mensajes iniciales que tienen menos de 200 caracteres de los 4242 que se escribieron, esto es el 91.8 % de los mensajes iniciales. Según los datos mostrados en la Figura 3.9, hay indicios de que los estudiantes cuando crean una consulta no tienden a escribir tanto como aquellas personas que van a participar posteriormente de la conversación. Esto se puede deber a que el mensaje inicial es usualmente una pregunta corta, mientras que los mensajes no iniciales suelen ser respuestas, que son más largas. En muchos casos el mismo hecho de que los estudiantes están solicitando ayuda en el foro de consultas implica que no tienen mucho para decir más que solicitar ayuda y explicar brevemente sus dudas, sumado a que es común que en las conversaciones de las consultas, los mentores y estudiantes traten de ayudar al creador de la consulta mostrando fragmentos de código, lo que hace que los mensajes no iniciales sean más extensos. Para la Figura 3.9

se consideraron a los mensajes de más de mil caracteres como parte de un mismo grupo ya que representan un porcentaje bajo del total de mensajes.

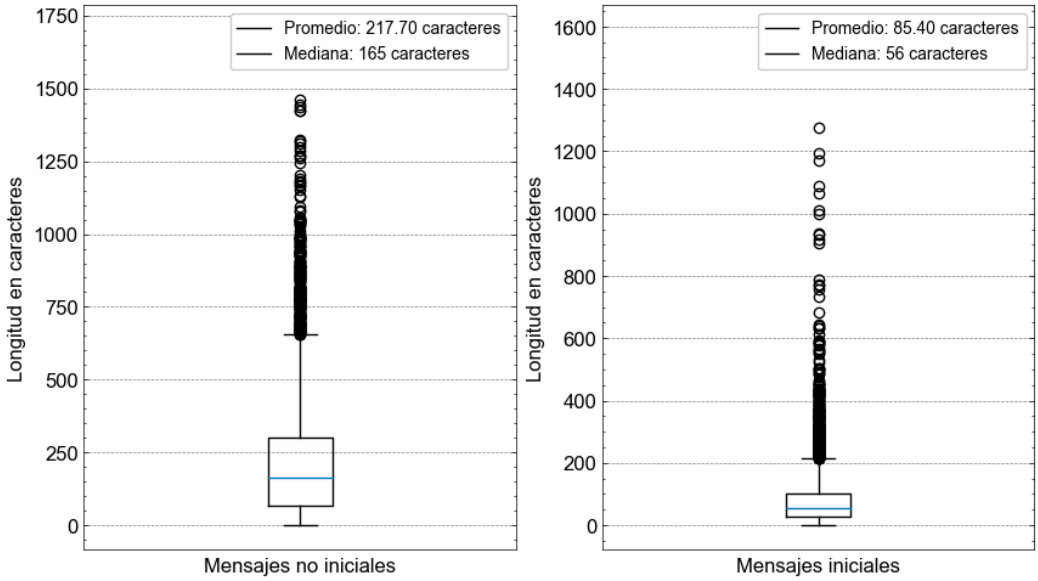


Figura 3.10: Boxplot: longitud de mensajes en el foro de consultas de Mumuki

La Figura 3.10 muestra la misma distribución en un bloxplot. Del lado izquierdo tenemos el boxplot de la longitud en caracteres de los mensajes, donde obtenemos una mediana de 165 caracteres, es decir, el 50 % de los mensajes tiene a lo sumo 165 caracteres y además el promedio de longitud es de 217.70 caracteres. Mientras tanto, del lado derecho tenemos el boxplot de los mensajes iniciales de las consultas, donde obtenemos una mediana de 56 caracteres, es decir, el 50 % de los mensajes tiene a lo sumo 56 caracteres. Además, el promedio de longitud es de 85.40 caracteres.

A partir de las tablas de mensajes y consultas también podemos analizar el largo de las conversaciones que se dieron en las consultas. En la Figura 3.11 se puede ver del lado izquierdo el histograma correspondiente a la cantidad de mensajes que se enviaron en las consultas. La conversación más larga tuvo 16 mensajes. No existen consultas sin mensajes ya que siempre existe por lo menos el mensaje inicial. Hay 1301 consultas en las que las conversaciones tuvieron un único mensaje (el mensaje inicial) representando el 30.6 % del total de las consultas. Estos casos implican que el estudiante que inició la consulta nunca recibió respuesta, lo cual muchas veces puede darse porque el estudiante resolvió el ejercicio luego de escribir la consulta, por lo que la consulta pasó a estado “Cerrada”. Si sumamos las consultas donde las conversaciones tuvieron entre 2 y 3 mensajes, suma un total 2439 consultas, representando el 57.4 % de total de las consultas.

Del lado derecho de la Figura 3.11 podemos ver el boxplot del largo de las conver-

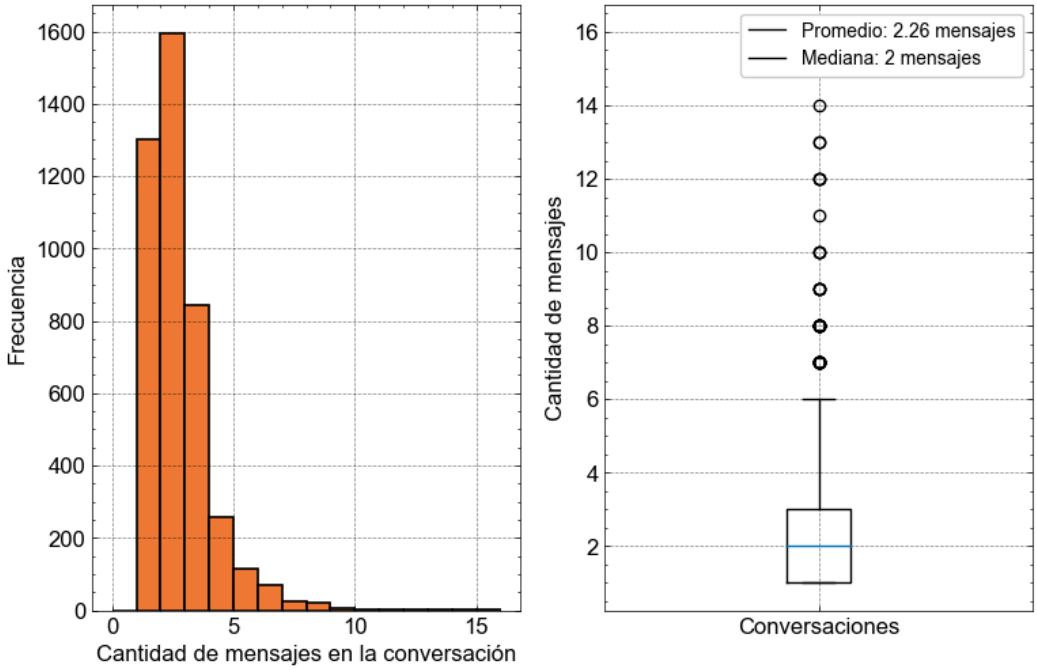


Figura 3.11: Largo de conversaciones en los foros de Mumuki

saciones, donde obtenemos que la mediana es de 2 mensajes, esto implica que el 50 % de las conversaciones tienen a lo sumo 2 mensajes. Además, el promedio de cantidad de mensajes en una conversación es de 2.26 mensajes.

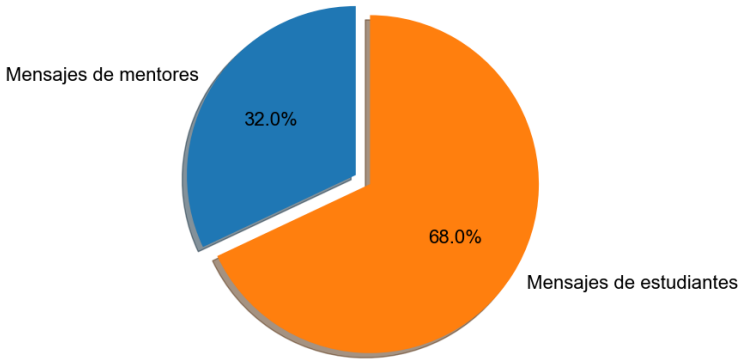


Figura 3.12: Participación de mentores y estudiantes en las conversaciones

A partir de los datos de mensajes y de usuarios que participaron en las conversaciones enviando los mensajes, podemos también ver qué porcentaje de los mensajes fueron enviados por mentores y estudiantes. Del total de 5324 mensajes, 1704 fueron enviados por mentores y 3620 fueron enviados por estudiantes, en la Figura 3.12 se puede ver esto gráficamente, donde los mensajes enviados por mentores represen-

tan el 32 % del total de mensajes enviados, mientras que los mensajes enviados por estudiantes representan el 68 %.

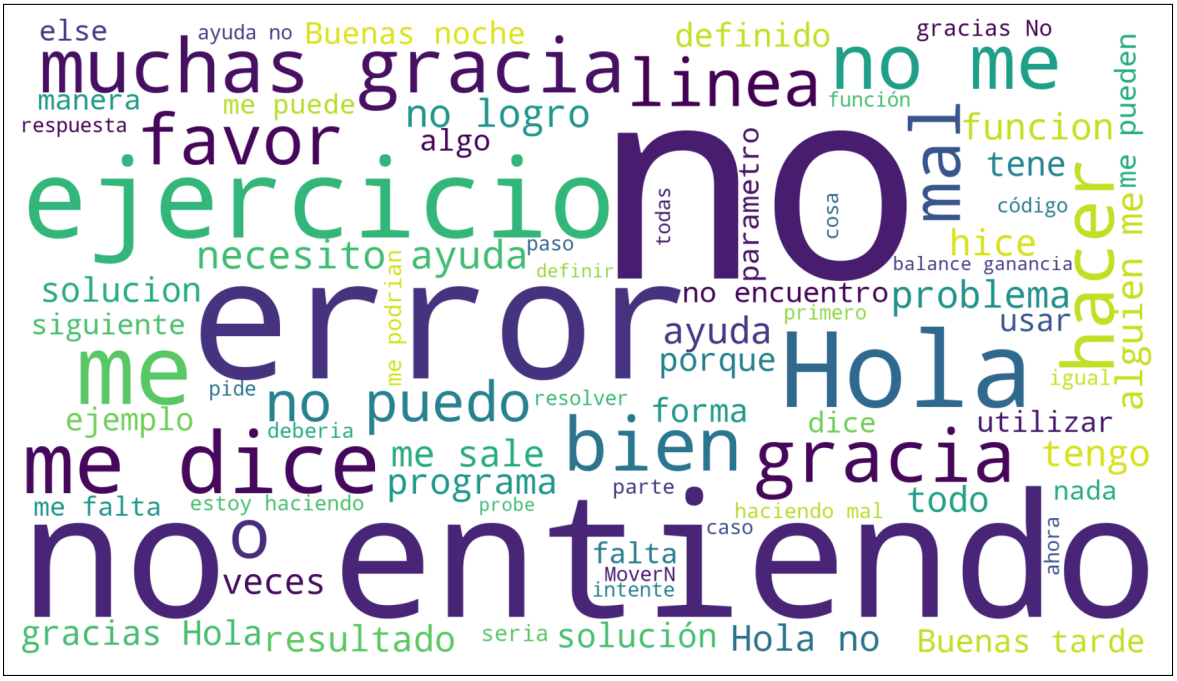


Figura 3.13: Palabras y frases más frecuentes escritas en el foro de consultas por los estudiantes

Con los datos de mensajes también podemos hacer un análisis de las palabras más escritas por los participantes del foro de consultas. En la Figura 3.13 se puede ver una nube de palabras correspondiente a las palabras o frases que aparecen con mayor frecuencia en los mensajes de estudiantes. Se pueden ver que las palabras o frases que aparecen con mayor frecuencia son *"no"*, *"ejercicio"*, *"error"*, *"no entiendo"*, *"problema"*, *"me dice"*. También aparecen otras palabras como *"gracias"*, *"hola"*, *"buenas noches"*, que cumplen una función social del lenguaje y que pueden ser frecuentes en backward looking acts.

La mayoría de las palabras en la Figura 3.13 parecen provenir de un contexto de desentendimiento y duda, lo cual respalda el hecho de que la longitud de los mensajes iniciales (que son escritos por estudiantes) sea menor a la de los demás mensajes: los estudiantes no son extensos al plasmar sus dudas.

Para poder ver la diferencia entre los mensajes que escriben los estudiantes y los mentores, podemos crear la misma nube pero sólo de frases y palabras presentes en los mensajes de mentores. En la Figura 3.14 podemos ver esta nube, donde se puede ver que las palabras utilizadas por los mentores provienen de un contexto de intento de explicar pasos a seguir para solucionar los ejercicios. Por ejemplo, se usan palabras como *"idea"*, *"usar"*, *"escribir"*, verbos en imperativo como *"fijate"*, *"acordate"*, y

verbos en condicional como *"tendría"*, *"debería"*, *"habría"*. A partir de esta nube de palabras se puede notar la diferencia de vocabulario que existe entre estudiantes y mentores, y la diferencia de intenciones que tienen al expresarse en los mensajes que escriben en el foro de consultas.

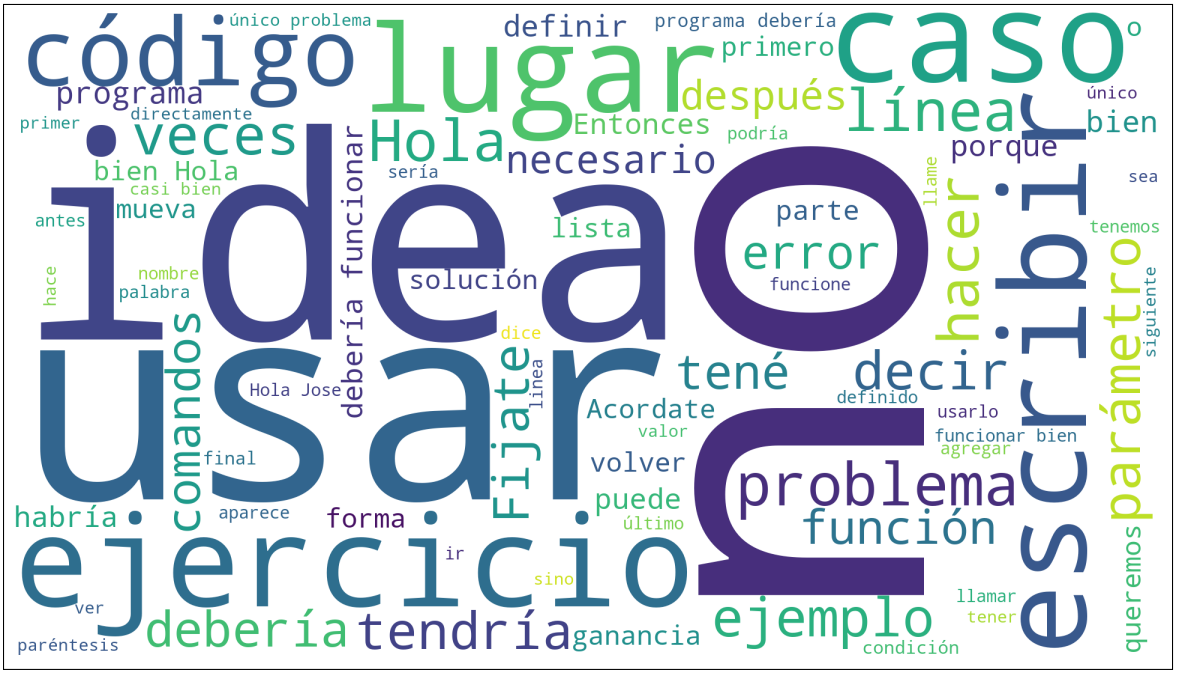


Figura 3.14: Palabras y frases más frecuentes escritas en el foro de consultas por los mentores

3.5. Relación con otros capítulos

En este capítulo presentamos el foro de consultas y los datos recolectados por el sistema online Mumuki, introducido como MOOC en el Capítulo 2. Detallamos el funcionamiento del sistema, la organización de los planes de estudio, cómo ve el usuario un ejercicio y cómo puede intentar resolverlo. Cuando el estudiante no puede resolverlo, el estudiante puede acceder al foro de consultas, cuyo funcionamiento analizamos además de los estados que pueden tener las consultas y las formas de transicionar entre estados. Luego, estudiamos estrategias implementadas por Mumuki para intentar detectar aquellas consultas donde hayan mensajes de estudiantes que requieran la atención de mentores. Por último, detallamos el conjunto de datos provisto por Mumuki y realizamos una descripción preliminar del mismo. En el Capítulo 4 veremos distintas etapas de procesamiento sobre el conjunto de datos como anonimización y anotación de los mensajes.

Capítulo 4

Metodología

En este capítulo se analiza la metodología utilizada para trabajar sobre el conjunto de datos (en inglés: *dataset*) disponible el cual fue descrito en el Capítulo 2. En la Sección 4.1 se describe el proceso de anonimización de los datos, luego en la Sección 4.2 se detalla el proceso de anotación de los mensajes en *forward y backward looking acts* y por último en la Sección 4.3 se realiza un análisis de características de los mensajes, las cuales se pudieron percibir mediante el proceso de anotación descrito en la Sección 4.2.

4.1. Anonimización

En esta sección se detalla el proceso de anonimización que se realizó sobre el conjunto de datos, con el fin de evitar exposición de datos de los usuarios en caso de que el conjunto de datos pudiera llegar a estar disponible públicamente en internet o en manos de un tercero no autorizado.

La privacidad es un derecho humano fundamental según el artículo 12 de la Declaración Universal de los Derechos Humanos y está protegida por múltiples marcos legales nacionales e internacionales, como la General Data Protection Regulation (GDPR) en Europa introducida en 2008 o por la Ley 25.326 de Protección de Datos Personales en el caso de Argentina. Todas estas normas restringen el uso y distribución de los datos como emails, casos judiciales o registros médicos, con el fin de proteger la privacidad de los datos personales. En el trabajo de Pierre et al. [25] se enfocan en el hecho de que muchas veces los datos son valiosos particularmente para investigaciones científicas, por lo que se necesitan usar los datos respetando la privacidad de los usuarios registrados en el conjunto de datos. Una solución para este problema es utilizar **técnicas de anonimización**, la cual consiste en remover o enmascarar cualquier información que directa o indirectamente puedan llevar a que un individuo sea identificado. En el caso de campos de datos estructurados la anonimización es

más simple que en el caso de no estructurados como texto libre, lo cual lleva a que se desarrollen distintos frameworks diseñados para extender la anonimización al texto libre. En [25] se presentan distintos enfoques para resolver el problema, y se hace hincapié en dos desafíos importantes a la hora de anonimizar los datos: primero, intentar que la anonimización sea robusta contra diferentes tipos de inferencias semánticas, basado en la información que un posible adversario podría poseer; y segundo, transformar el texto minimizando el riesgo de divulgar información personal y mantener la semántica tanto como sea posible.

En el caso de nuestro trabajo tenemos un conjunto de datos donde hay campos estructurados y no estructurados como el texto de los mensajes que se envían en una conversación de una consulta. En particular, este último campo es el que mayor desafío presenta ya que contiene texto libre, entonces es muy probable que la gente filtre información personal explícita o implícitamente, por ejemplo escribiendo nombres completos, números de teléfono, direcciones, emails, etc., por lo que es importante analizar y anonimizar estos datos con cuidado.

La anonimización del conjunto de datos fue realizada por Franco Bulgarelli, COO de la empresa desarrolladora de Mumuki. Como primer intento, se utilizó spaCy[1], una librería de código abierto para procesamiento de lenguaje natural en Python. Dentro de spaCy existen distintos proyectos, uno de ellos es Presidio[2], el cual permite anonimizar texto reemplazando datos sensibles por etiquetas. En la Tabla 4.1 se presenta un ejemplo de anonimización mostrado en la página web de Presidio.

Texto de input	**Texto de output**
Please cancel my credit card effective September 19th. My name is Aarav Navuluri and my credit card number is 4095-2609-9393-4932. My email is aatav@presidio site and I live in Amherst.	Please cancel my credit card effective <DATE TIME>. My name is <PERSON> and my credit card number is <CREDIT CARD>. My email is <EMAIL ADDRESS> and I live in <LOCATION>.

Tabla 4.1: Ejemplo de anonimización de Presidio

En la Tabla 4.1 se puede ver del lado izquierdo el input que se le da a Presidio, un texto donde hay gran cantidad de datos personales: nombre completo de la persona, número de tarjeta de crédito, número de teléfono, e-mail, localidad. Y del lado derecho se ve el output que Presidio devuelve, anonimizando cada uno de estos datos con etiquetas correspondientes al dato sensible del cual se trata.

[1] https://spacy.io
[2] https://spacy.io/universe/project/presidio

A pesar de intentar utilizar Presidio para anonimizar todas las entradas de texto libre en nuestro conjunto de datos, el principal problema de Presidio es que no funciona bien con el idioma español, dado que fue desarrollado para anonimizar textos en inglés como se puede ver en el ejemplo mostrado en la Tabla 4.1. Por esto, se decidió realizar reemplazos utilizando expresiones regulares. Para reemplazar los números de teléfono, números de documento y e-mails se utilizaron expresiones regulares para detectarlas mientras que los nombres de personas fueron detectados siguiendo los nombres de todas las personas registradas que participaron del plan de estudio en Mumuki. Una vez detectados los datos sensibles, se los reemplazó con datos falsos generados automáticamente con la librería faker[3] de Python. También se trataron los caracteres unicode con la librería unidecode[4]. Por último, para anonimizar los datos estructurados se utilizó la librería pandas[5], que permite mediante su función *factorize* codificar los campos, manteniendo la consistencia para poder seguir haciendo uso de las relaciones entre tablas. Luego del proceso automático, se realizó un proceso manual para refinar y finalizar el proceso de anonimización.

En la siguiente sección se detalla el proceso de anotación de los mensajes del conjunto de datos en forward o backward looking acts: estas anotaciones luego serán útiles para poder realizar experimentos con modelos de aprendizaje automático.

4.2. Anotación

En esta sección describimos el proceso de anotación de los mensajes enviados en el foro, presentes en el conjunto de datos disponible, clasificándolos en forward o backward looking acts. En la Sección 4.2.1 definimos qué subconjunto de mensajes del conjunto de datos anotar, en la Sección 4.2.2 definimos consideraciones que se tuvieron en cuenta a la hora de anotar para tener un común acuerdo entre anotadores y calculamos el Coeficiente kappa de Cohen para estimar el nivel de acuerdo entre ellos, luego en la Sección 4.2.3 detallamos el sistema utilizado para la anotación conjunta, y por último en la Sección 4.2.4 mostramos los resultados obtenidos del proceso de anotación.

Nos referiremos a los mensajes enviados en una consulta (tanto el mensaje inicial como no los mensajes no iniciales) como *discusión*. Sabemos, por lo visto en el Capítulo 3, que Mumuki cuenta con el marcador "no es una pregunta", éste último con la finalidad determinar que, si el último mensaje de una discusión es de un estudiante y está marcado como una pregunta, que es la opción por defecto, entonces la discusión

[3] https://faker.readthedocs.io/en/master/
[4] https://pypi.org/project/Unidecode/
[5] https://pandas.pydata.org

tiene un mensaje que requiere atención de un mentor. Debido a que el objetivo de este trabajo es poder identificar si un mensaje es un forward o backward looking act, la negación del marcador "no es una pregunta" se puede interpretar como un intento de aproximación manual por parte de Mumuki a nuestro a objetivo. Aún así, como "no es una pregunta" es una opción introducida recientemente en Mumuki al momento de analizar el conjunto de datos, no existe una anotación manual correcta y completa del mismo, por lo que debemos abordar la anotación de cada uno de los mensajes, clasificándolos en forward o backward looking act.

En la siguiente sección se presenta el detalle del proceso llevado a cabo para determinar qué conjunto de datos se anotará.

4.2.1. Definición del conjunto de datos a anotar

Como se analizó en el Capítulo 3, en el conjunto de datos proporcionado por Mumuki se tienen 5324 mensajes en total distribuidos en 4242 discusiones diferentes. Cuando un estudiante realiza una consulta, inicia una discusión con un mensaje inicial presente en el campo *description* de la tabla de discusiones. Debido a que la descripción de una consulta es el mensaje inicial de la discusión, se decide considerarlos como mensajes, por lo que en total tenemos 9566 mensajes para anotar.

Para realizar la anotación se decidió **no anotar aquellos mensajes que hayan sido enviados por mentores**: ya que la finalidad de los mismos es brindar ayuda a los estudiantes, probablemente el mensaje de un mentor no sea un forward looking act, sumado al hecho de que clasificar el mensaje de un mentor como que requiere la atención de un mentor es en cierto sentido absurdo, ya que un mentor no puede prestarle atención a su propio mensaje ya que él lo escribió, y además se asume que un mentor no acude ni le pide ayuda a otro mentor. Quitando los mensajes enviados por mentores en todas las discusiones, queda un conjunto de 7861 mensajes restantes para anotar.

Luego, realizando una observación manual de todos los **mensajes iniciales de las discusiones**, se decidió considerarlos a todos como forward looking acts ya que son mensajes de estudiantes explicando sus dudas, solicitando ayuda, expresando desentendimiento, confusión y en algunos casos frustración con respecto a la resolución del ejercicio, por lo que se considera que es muy poco probable que el mensaje inicial de una discusión no sea un mensaje de este tipo. Quitando los mensajes iniciales de las discusiones, queda un conjunto final de 3620 mensajes para anotar.

En la Figura 4.1 se puede ver cómo el proceso de decisión de qué mensajes anotar manualmente nos sirvió para reducir el conjunto de mensajes totales a anotar, facilitando el trabajo de anotación y permitiendo llevarlo a cabo en un menor tiempo.

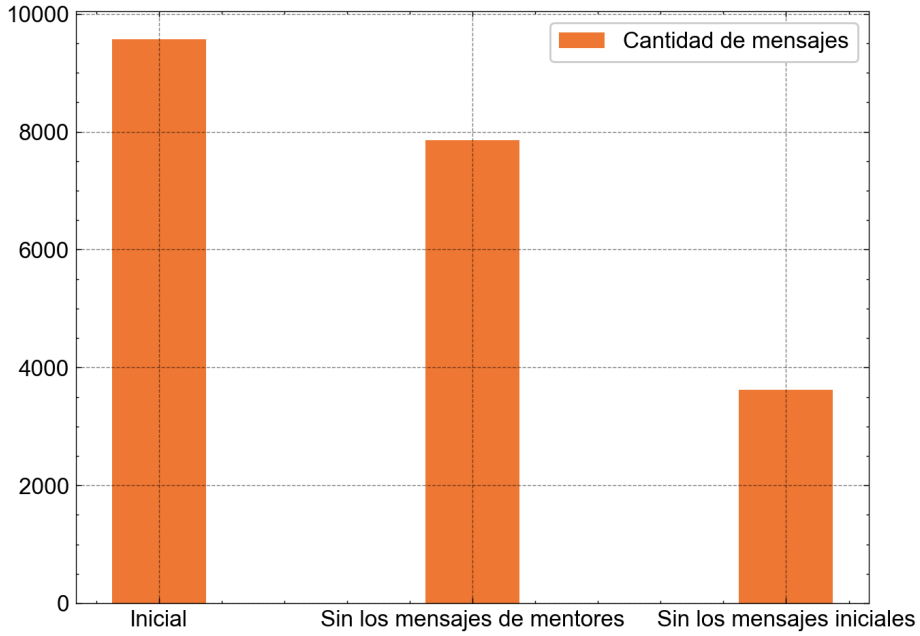


Figura 4.1: Filtrado del conjunto de mensajes a anotar

La cantidad de mensajes originalmente eran 9566, el cual se puede ver en la barra "Inicial" a la izquierda, luego, en la barra del centro "Sin los mensajes de mentores" se pueden ver representados los 7861 mensajes que quedan luego de quitar los mensajes enviados por mentores, representando una reducción del 17 % del conjunto de mensajes a anotar. Por último, en la barra de la derecha "Sin los mensajes iniciales" se representan los 3620 mensajes finales a anotar que quedan luego de quitar los mensajes de mentores y los mensajes iniciales, ya que estos últimos fueron considerados forward looking acts obteniendo una reducción del 62 % del conjunto de mensajes a anotar.

En la sección siguiente se detallan las consideraciones en común entre los anotadores previo al proceso de anotación, y se calcula el nivel de acuerdo entre ellos.

4.2.2. Nivel de acuerdo entre anotadores y consideraciones en común

En la anotación de los 3620 mensajes participaron tres anotadores: Mariano Piatti, el autor de esta tesis, Marcos Gómez, el director de la tesis; y Franco Bulgarelli, COO de la empresa desarrolladora de Mumuki.

Previo a la anotación, se pautaron acuerdos con respecto a dos casos específicos de mensajes y ambos tienen que ver con la participación de otros estudiantes en una discusión. El primer caso es que si el mensaje a clasificar es de un estudiante diferente al que inició la discusión, y en el mismo intenta responder la consulta o ayudar, entonces se lo considera un backward looking act por el hecho de responder

una duda. El segundo caso es similar, solo que en este caso el otro estudiante suma dudas a la consulta o expresa su imposibilidad de resolverlo con mensajes de la forma *"A mí tampoco me sale"*, entonces se lo considera como forward looking act.

Con estas consideraciones en común, sumado al conocimiento de los conceptos de forward y backward looking acts, los tres anotadores realizaron 87 anotaciones sobre los mismos mensajes con el fin de calcular el Coeficiente kappa de Cohen (κ)[6] y así poder estimar el nivel de acuerdo entre ellos. $\kappa = 1$ si todos los anotadores están de acuerdo y $\kappa = 0$ si no hay acuerdo distinto al que se espera que habría por azar. Es importante aclarar que este coeficiente sirve para medir a dos anotadores distintos, y como en este trabajo de anotación contamos con tres anotadores, calcularemos tres coeficientes: uno para cada par de anotadores. De las anotaciones obtuvimos los resultados mostrados en la Tabla 4.2.

Anotador	Forward looking act **(F)**	Backward looking act **(B)**
Mariano	58	29
Marcos	59	28
Franco	61	26

Tabla 4.2: Anotaciones realizadas por cada anotador

Considerando F = Forward looking act, B = Backward looking act, en la Tabla 4.2 podemos ver que Mariano anotó 58 mensajes como F, y 29 como B, Marcos anotó 59 mensajes como F y 28 como B, y Franco anotó 61 mensajes como F y 26 como B. Sumado a esto, para calcular el Coeficiente kappa de Cohen es necesario tener en cuenta las coincidencias entre anotadores.

En la Tabla 4.3 podemos ver las coincidencias entre los anotadores, en la primera tabla vemos las coincidencias para Mariano y Franco, donde coincidieron en 58 F, 26 B y discreparon en 3 mensajes donde Mariano los marcó como B y Franco como F. En la segunda tabla se ven coincidencias entre Marcos y Franco, coincidiendo en 58 F y 25 B, discrepando en 3 mensajes donde Marcos las marcó como B y Franco como F, y en un mensaje el cual Marcos marcó como F y Franco como B. Por último en la tercera tabla se ven las coincidencias entre Marcos y Mariano, donde coincidieron en 57 F, 27 B y discreparon en un mensaje que Marcos marcó como B y Mariano como F, y en dos mensajes que Marcos marcó como F y Mariano como B.

$$\kappa = \frac{Pr(a) - Pr(e)}{1 - Pr(e)} \tag{4.1}$$

En la Ecuación 4.1 tenemos la fórmula del Coeficiente de kappa de Cohen, donde

[6] https://es.wikipedia.org/wiki/Coeficiente_kappa_de_Cohen

	Franco	
	Forward looking act	Backward looking act
Mariano	Forward looking act	58	0
	Backward looking act	3	26

	Franco	
	Forward looking act	Backward looking act
Marcos	Forward looking act	58	1
	Backward looking act	3	25

	Mariano	
	Forward looking act	Backward looking act
Marcos	Forward looking act	57	2
	Backward looking act	1	27

Tabla 4.3: Coincidencias entre grupos de dos anotadores

$Pr(a)$ es el acuerdo relativo entre los dos anotadores y $Pr(e)$ es la probabilidad de acuerdo entre ellos por azar. Utilizando la Tabla 4.3, en las Ecuaciones 4.2, 4.3 y 4.4 calculamos las probabilidades de elegir F y B de cada anotador.

$$P_F(Franco) = 0.7011 \quad P_B(Franco) = 0.2989 \tag{4.2}$$

$$P_F(Marcos) = 0.6781 \quad P_B(Marcos) = 0.3218 \tag{4.3}$$

$$P_F(Mariano) = 0.6666 \quad P_B(Mariano) = 0.3333 \tag{4.4}$$

Con todo esto, calculamos el coeficiente para los anotadores Franco y Mariano en las Ecuaciones 4.5, 4.6 y 4.7.

$$Pr(a) = \frac{\#concordancias}{\#anotaciones} = \frac{58 + 26}{87} = 0.9655 \tag{4.5}$$

$$Pr(e) = P_F(Franco) * P_F(Mariano) + P_B(Franco) * P_B(Mariano) = 0.5669 \tag{4.6}$$

$$\kappa = \frac{Pr(a) - Pr(e)}{1 - Pr(e)} = \mathbf{0.89725} \tag{4.7}$$

Luego, calculamos el coeficiente para los anotadores Franco y Marcos en las Ecuaciones 4.8, 4.9 y 4.10.

$$Pr(a) = \frac{\#concordancias}{\#anotaciones} = \frac{58+25}{87} = 0.9540 \tag{4.8}$$

$$Pr(e) = P_F(Franco) * P_F(Marcos) + P_B(Franco) * P_B(Marcos) = 0.5715 \tag{4.9}$$

$$\kappa = \frac{Pr(a) - Pr(e)}{1 - Pr(e)} = \mathbf{0.8926} \tag{4.10}$$

Y por último, calculamos el coeficiente para los anotadores Marcos y Mariano en las Ecuaciones 4.11, 4.12 y 4.13.

$$Pr(a) = \frac{\#concordancias}{\#anotaciones} = \frac{57+27}{87} = 0.9655 \tag{4.11}$$

$$Pr(e) = P_F(Mariano) * P_F(Marcos) + P_B(Mariano) * P_B(Marcos) = 0.5592 \tag{4.12}$$

$$\kappa = \frac{Pr(a) - Pr(e)}{1 - Pr(e)} = \mathbf{0.9217} \tag{4.13}$$

Se puede ver que los Coeficientes de kappa de Cohen calculados para cada par de anotadores en las Ecuaciones 4.7, 4.10 y 4.13 son valores cercanos a 1, por lo que los anotadores tienen un nivel de acuerdo alto.

En la sección a continuación se detalla el sistema online implementado para llevar a cabo la anotación de los mensajes del conjunto de datos.

4.2.3. Sistema de anotación

Para realizar la anotación conjunta de todos los mensajes, se diseñó un sistema online de anotación, al cual cada anotador accedía, indicaba su nombre y comenzaba el proceso de anotación. El proceso de anotación consistía en visualizar un mensaje por vez, y para cada mensaje seleccionar una de dos opciones: si es un forward looking act; o si es un backward looking act. Una vez que el anotador seleccionaba una opción, el sistema pasaba a mostrar el siguiente mensaje a anotar.

Debido a que decidir si un mensaje es un forward o backward looking act no es una tarea directa teniendo únicamente el texto del mensaje a clasificar, se decidió que

el sistema muestre el mensaje a clasificar junto con todos los mensajes previos que habían sido enviados en la conversación, incluyendo los mensajes que fueron enviados por mentores, los cuales eran marcados con una etiqueta especial que los distingue. También, cada mensaje mostraba un identificador de la persona que lo envió, para poder distinguir los momentos en que una persona participó diferentes veces en una misma discusión. El hecho de mostrar toda la discusión previa e identificar a las personas que participaron en ella sirve para poder brindarle al anotador el contexto del mensaje, con el fin de lograr mayor precisión al anotar. Por ejemplo, si tuviéramos que clasificar el mensaje *"Exactamente. Ese es el motivo por el que pido ayuda para solucionar el ejercicio."* sin ninguna información del contexto en el que se envió, no sabemos si el mensaje fue enviado al final de una discusión donde la consulta ya está resuelta pero el estudiante aclara por qué pidió ayuda originalmente, o si el mensaje fue enviado en el inicio de una discusión y está respondiendo una pregunta clarificadora de un mentor para poder entender mejor la duda del estudiante.

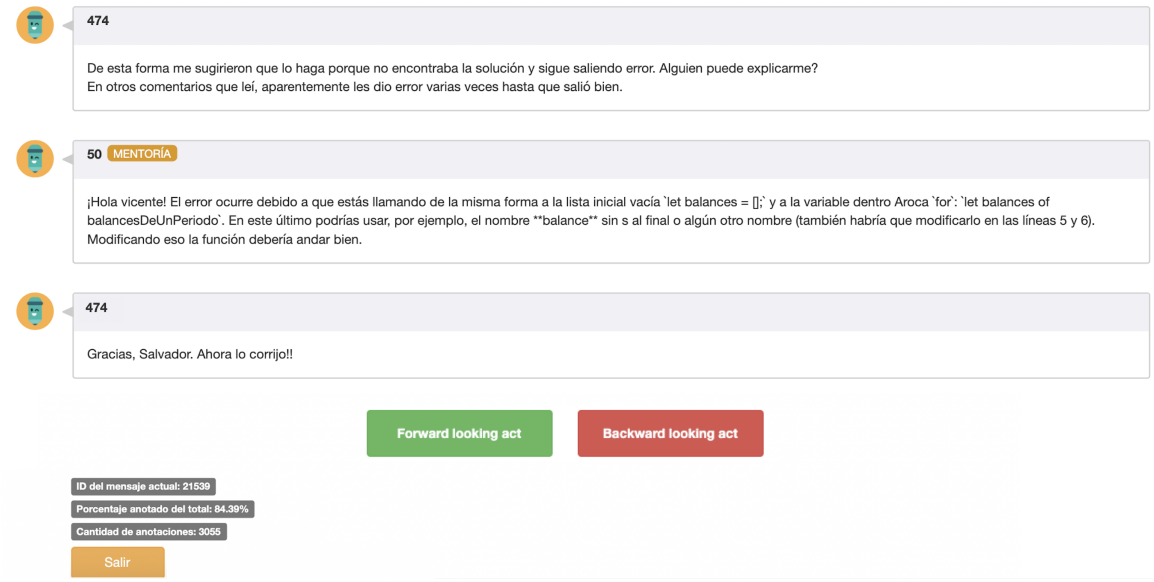


Figura 4.2: Sistema de anotación

En la Figura 4.2 podemos ver un ejemplo de anotación en el sistema. Se pueden observar tres mensajes: el primer mensaje es el que envió el estudiante que inició discusión, en este caso lo envió el estudiante que es representado con el identificador 474. Aquí se puede ver un ejemplo que refleja el motivo por el cual consideramos los mensajes iniciales como forward looking acts, el mensaje inicial del estudiante 474 expresa confusión y espera una ayuda por parte de un mentor. El segundo mensaje tiene junto al identificador un cartel en amarillo con la leyenda "MENTORÍA", lo cual

indica que el mensaje fue enviado por un mentor. En este mensaje se puede ver que el mentor que es representado con el identificador 50 intenta responder a la consulta y ayudar al estudiante 474, lo cual también es un ejemplo que refleja la decisión de no anotar los mensajes de mentores ya que muy probablemente son backward looking acts. El tercer mensaje es una respuesta del estudiante 474 donde agradece la ayuda al mentor 50.

En el sistema de anotación, el mensaje a clasificar es el último mensaje de la discusión mostrada. En el caso de la Figura 4.2 el mensaje a anotar es el tercer mensaje, es decir, la contestación del estudiante 474 al mentor 50. Para anotar y decidir si el mensaje es un forward o backward looking act, el sistema muestra dos botones debajo del último mensaje: "Forward looking act" en verde, o "Backward looking act" en rojo, los cuales anotan al mensaje como tales. También se pueden ver debajo de los botones de anotación tres datos relacionados con el proceso de anotación hasta el momento: Un identificador del mensaje que se va a anotar, el porcentaje anotado hasta el momento del total de mensajes y la cantidad de anotaciones que el anotador realizó hasta el momento. Debajo de estos, hay un botón amarillo con la leyenda "Salir" que permite al anotador salir del sistema. Para continuar con el ejemplo de anotación, en el caso de la Figura 4.2, el mensaje a anotar es un mensaje de agradecimiento, donde el estudiante 474 da a entender que entendió la ayuda del mentor 50 y que va a proceder a solucionar el ejercicio. Como el mensaje no tiene la intención de que el mentor responda, el mensaje debe ser marcado como un backward looking act.

4.2.4. Resultados de la anotación

Del total de 3620 mensajes a anotar, obtuvimos que 866 mensajes fueron anotados como forward looking acts y 2754 como backward looking acts. Esto se puede ver en la Tabla 4.4.

Forward looking acts	866 mensajes
Backward looking acts	2754 mensajes

Tabla 4.4: Resultados de la anotación de mensajes

También se puede realizar una distinción sobre los mensajes iniciales ya que se consideran forward looking acts a priori, descartándolos de la anotación manual. Podemos ver esta distinción en la Figura 4.3 donde del lado izquierdo tenemos la proporción de mensajes anotados, obtenido a partir de la Tabla 4.4, donde el 76 % de los mensajes son backward looking acts y el 24 % restante son forward looking acts.

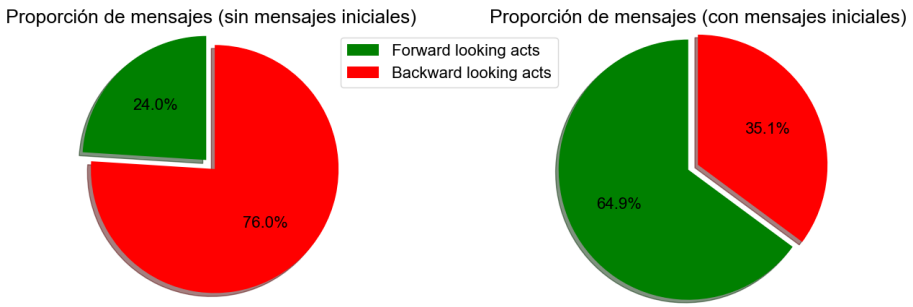


Figura 4.3: Proporción de forward y backward looking acts en mensajes anotados

Como es de interés analizar los mensajes de todos los estudiantes, del lado derecho calculamos la proporción teniendo en cuenta los mensajes iniciales también, que son 4242 mensajes clasificados como forward looking act. De esta manera, y como se puede ver en la Tabla 4.5 tenemos 7861 mensajes de estudiantes, de los cuales 5107 son forward looking act y los restantes 2754 son backward looking act, representando el 64.9 % y 35.1 % de los mensajes respectivamente.

Forward looking act	5107 mensajes
Backward looking act	2754 mensajes
Total	7861 mensajes

Tabla 4.5: Clasificación final de mensajes de estudiantes

De esta manera, del proceso de anotación se puede concluir que una vez comenzada una discusión en el foro de consultas de Mumuki, es más probable que cualquier mensaje enviado por un estudiante sea un backward looking act. Esto implica que cada vez que un estudiante participe de una discusión en el foro, es más probable que el mensaje que envió el estudiante no requiera atención a que si la requiera. Esto demuestra la importancia de poder contar con un sistema automático de detección de mensajes que requieran atención, ya que aproximadamente el 75 % de los mensajes en una consulta no requieren la atención de un mentor y representan una pérdida de tiempo de los mismos identificando aquellos mensajes que sí necesiten atención. También, al considerar los mensajes iniciales, se tiene un conjunto de datos donde aproximadamente el 65 % de los mensajes son forward looking acts, lo cual es útil para llevar a cabo el entrenamiento de clasificadores automáticos y poder detectar los forward looking acts presentes en los mensajes de las consultas.

En la sección siguiente se describen características propias de los mensajes forward y backward looking acts.

4.3. Descripción de características

Luego de la anotación surge la necesidad de poder encontrar características que diferencian a aquellos mensajes que son forward looking acts de aquellos que son backward looking acts. En esta sección analizamos distintas características que luego serán útiles para el entrenamiento de modelos de aprendizaje automático.

Longitud del mensaje

La longitud del mensaje en caracteres proporciona información estadística: como se vió anteriormente en el Capítulo 3, es común que los estudiantes que envían un mensaje que es forward looking act no escriban muchos caracteres en el contenido del mensaje, simplemente piden ayuda y/o explican brevemente su duda. En cambio, cuando un estudiante envía un mensaje que es un backward looking act, suele contener muchos más caracteres en comparación, ya sea porque es un texto largo y detallado que busca resolver la duda del estudiante creador de la consulta, o contiene fragmentos de código.

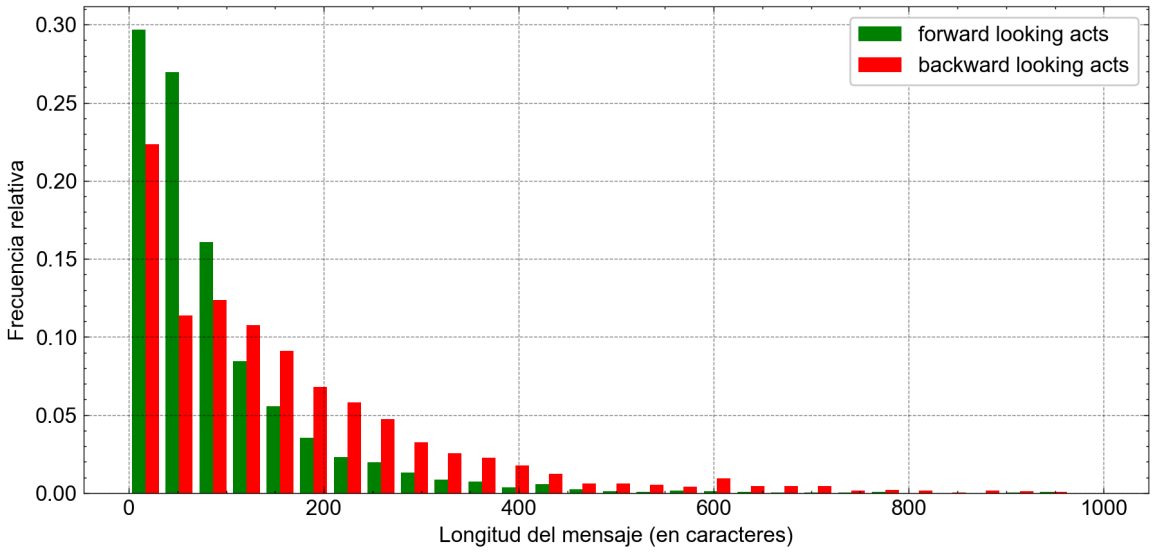


Figura 4.4: Longitud de mensajes conteniendo backward y forward looking acts

En la Figura 4.4 se presenta un histograma que compara las longitudes de los mensajes, tanto para aquellos que son forward looking acts como los que son backward looking acts. Se puede ver que los mensajes que contienen hasta 100 caracteres representan una mayor proporción de los forward looking acts que de los backward looking acts, y a partir de los 100 caracteres este comportamiento se invierte. Esto también coincide con el hecho de que el promedio de caracteres de mensajes que son

forward looking acts es de 94.93 caracteres, mientras que el promedio para los que son backward looking acts es de 160.42 caracteres.

Bolsa de palabras: forward looking acts

Como se vió en el Capítulo 3, existen distintas palabras y frases que ocurren frecuentemente en los mensajes enviados en el foro. En particular se pueden distinguir distintas frases que los estudiantes suelen escribir a la hora de buscar ayuda en el foro, por ejemplo una de las más comunes es *"no entiendo"*. Es por esto que podemos formar bolsas de palabras comunes y en caso de que alguna de estas ocurra en el texto del mensaje, la probabilidad de que el mismo sea un forward looking act es mayor. Para crear las bolsas de palabras formaremos N-gramas [7] de una, dos y tres palabras, llamados unigramas, bigramas y trigramas respectivamente. Los N-gramas representan secuencias de palabras en un texto dado, por lo que en este caso son secuencias de palabras que ocurren en el texto del mensaje separados por un caracter "espacio". En la Tabla 4.6 podemos ver ejemplos de N-gramas para los mensajes que son forward looking acts, todos ellos son frases y palabras que expresan duda y desentendimiento como por ejemplo el unigrama *ayuda*, el bigrama *no entiendo* o el trigrama *no me sale*, y si están presentes en un mensaje es porque probablemente el mensaje sea un forward looking act.

Unigramas	*ayuda, entendiendo, entender, explicar*
Bigramas	*no entiendo, mi duda, no comprendo*
Trigramas	*no me sale*

Tabla 4.6: N-gramas que representan a los mensajes que contienen forward looking acts

Esto se puede ver en la Figura 4.5: de los 7861 mensajes de estudiantes, 2598 contienen alguno de los N-gramas de la Tabla 4.6 y de ellos, 2461 fueron anotados como forward looking acts y los restantes 137 como backward looking acts, representando el 48.18 % y el 4.9 % de sus respectivas categorías, es decir, el 48.18 % de los mensajes que fueron anotados como forward looking acts contienen algún N-grama de la Tabla 4.6.

[7] https://es.wikipedia.org/wiki/N-grama

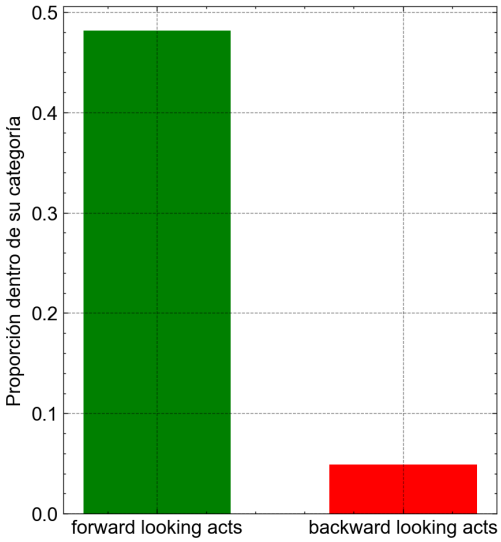


Figura 4.5: Proporción de mensajes que contienen alguno de los N-gramas de la Tabla 4.6

Bolsa de palabras: backward looking acts

También podemos dar N-gramas que sean comunes en los mensajes que son backward looking acts. En el Capítulo 3 se observó que los mentores generalmente escriben mensajes que son backward looking acts, junto con ejemplos de palabras y frases que utilizan al intentar responder las dudas de los estudiantes, las cuales también son representativas del lenguaje que un estudiante utilizaría al escribir un mensaje en el foro con la intención de no necesitar la atención de un mentor: ya sea ayudando a otro estudiante con dudas, agradeciendo, aclarando, etc. Comúnmente se utilizan verbos en condicional simple y en imperativo, y además, como se trata de un contexto de aprendizaje de programación, se suelen utilizar frases (generalmente verbos en tiempo pretérito) que representan que ahora el programa que resuelve el ejercicio ya funciona, o que la duda ya fue resuelta como por ejemplo: *ya funciona*.

Unigramas	*gracias, tendría, debería, habría, debes, tené, hacé, solucioné, entendí, arreglé, funcionó, agrega, acordate*
Bigramas	*ya funciona, lo logré, te olvidaste, te faltó, te falta*
Trigramas	*eso pasa porque*

Tabla 4.7: N-gramas que representan a los mensajes que contienen backward looking acts

También es importante tener en cuenta que es común que los backward looking acts contengan agradecimientos, generalmente a través de la palabra *gracias*. En la

Tabla 4.7 se pueden ver ejemplos de N-gramas representativos de los backward looking acts. El hecho de que alguno de ellos ocurra en un mensaje en el foro enviado por un estudiante implica que la probabilidad de que el mensaje sea un backward looking act es mayor, como se puede apreciar en la Figura 4.6: de los 7861 mensajes de estudiantes, 2338 contienen algún N-grama de la Tabla 4.7, y de ellos, 1391 fueron anotados como backward looking acts y 724 como forward looking acts, representando el 50.50 % y el 14.17 % de sus respectivas categorías, es decir, el 50.50 % de los mensajes que fueron anotados como backward looking acts contienen algún N-grama de la Tabla 4.7.

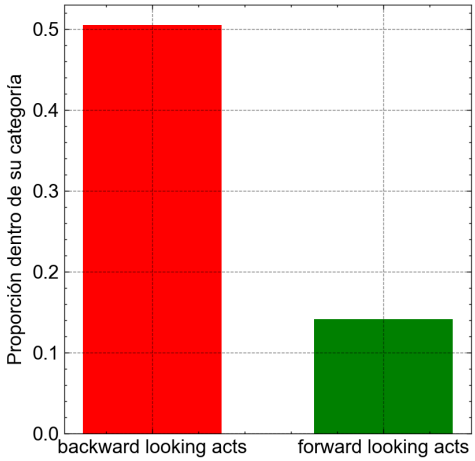


Figura 4.6: Proporción de mensajes que contienen alguno de los N-gramas de la Tabla 4.7

4.4. Relación con otros capítulos

En este capítulo analizamos el procesamiento realizado sobre el conjunto de datos disponible descrito en el Capítulo 3, empezando por el proceso de anonimización, donde se intentó utilizar la herramienta de detección automática de datos personales Prestigio, la cual al estar creada para ser utilizada en el idioma inglés, se terminó utilizando búsqueda por expresiones regulares y utilización de librerías en Python para el refinamiento y anonimización de los datos personales que puedan existir en los campos de texto libres del conjunto de datos, sumado a un trabajo manual. También se documentó el proceso de anotación llevado a cabo, para poder clasificar los mensajes enviados por los estudiantes en el foro en forward looking acts y backward looking acts, se detallaron distintas consideraciones previas a la anotación y comunes acuerdos entre los anotadores. Para poder llevarlo a cabo se diseñó un sistema de anotación online donde participaron tres anotadores, los cuales anotaron 3620 mensajes en total. Luego, se realizó un análisis estadístico de los resultados de la anotación y por último se detallaron las características presentes en los mensajes, las cuales fueron observadas

durante el proceso de anotación y permiten distinguir la categoría a la que un mensaje pertenece. En el siguiente capítulo se analiza cómo resolver el problema de determinar si un mensaje es un forward o backward looking act de manera automática y se presentan los modelos de aprendizaje automático con los cuales se experimentará. En el Capítulo 6 se construirán los modelos y se estudiarán sus desempeños.

Capítulo 5

Aprendizaje automático para clasificación de mensajes

En este capítulo se describe el diseño de los experimentos a utilizar con el fin de clasificar los mensajes de estudiantes en forward o backward looking act. En la Sección 5.1 se explica cómo la tarea puede ser modelada como un problema de clasificación de aprendizaje supervisado. En la Sección 5.2 se muestra cómo obtener el vector de características de un mensaje para ser utilizado en modelos de aprendizaje automático. En la Sección 5.3 se detallan las métricas que se utilizarán para poder medir qué tan bien funcionan los modelos de clasificación obtenidos en los experimentos. En la Sección 5.4 se detallan brevemente los modelos de aprendizaje automático que se utilizarán en los experimentos de este trabajo: regresión logística, bosque aleatorio y redes neuronales. En la Sección 5.5 se presenta la técnica de validación cruzada para mejorar la independencia de los resultados con respecto a la partición que se realice del conjunto de datos en prueba y entrenamiento.

5.1. Aprendizaje automático sobre mensajes

En esta sección se introducen brevemente y desde una perspectiva histórica los conceptos de inteligencia artificial, aprendizaje automático y el tipo de método que se utilizará en este trabajo: aprendizaje automático supervisado. También se describe cómo esta tarea puede ser modelada como un problema de aprendizaje automático supervisado.

La **inteligencia artificial** (IA) es una rama de las ciencias de la computación cuyo origen tiene tantos años como la computación misma y se da en la Tesis de Church-Turing [24], la cual sostiene que las computadoras (construidas a partir del concepto de máquina de Turing [48]) pueden simular cualquier proceso de razona-

miento formal. A partir de esto, es común cuestionarse el hecho de que una máquina pueda tener inteligencia, tema el cual es abordado por primera vez por Turing [47], donde plantea la pregunta: *¿Pueden pensar las máquinas?*. Como "pensar" es difícil de definir, Turing plantea una prueba llamada *Test de Turing* la cual consiste en que un interrogador humano debe intentar diferenciar entre un texto de respuesta generado por una máquina y por un humano. Si el humano no puede diferenciar entre ellos, entonces la máquina pasa la prueba.

Una de las ramas de la inteligencia artificial es el **aprendizaje automático**, que es el estudio de algoritmos que pueden mejorar automáticamente a través de la experiencia y el uso de datos [30]. Los algoritmos de aprendizaje automático construyen un modelo basado en datos simples con el fin de realizar predicciones o tomar decisiones sin estar explícitamente programados para ello. Los motores que impulsan el aprendizaje automático son los algoritmos, entre los cuales se pueden diferenciar distintos tipos y uno de ellos es el **aprendizaje supervisado**. El aprendizaje supervisado construye un modelo matemático a partir de un conjunto de datos que contienen tanto el input como el output deseados [41]. Este conjunto de datos es conocido como conjunto de **datos de entrenamiento** y consiste de ejemplos de entrenamiento: un ejemplo de entrenamiento contiene uno o más inputs y el output deseado. A través de una optimización iterativa de una función de costo [38], los algoritmos de aprendizaje supervisado aprenden una función que puede ser utilizada para predecir el output asociado a nuevos inputs [31]. Existen distintos tipos de algoritmos de aprendizaje supervisado, entre los cuales se encuentran los **algoritmos de clasificación**: estos algoritmos son utilizados cuando el output está restringido a un conjunto limitado de valores. Más aún, si el output está restringido únicamente a dos valores, se está trabajando sobre un problema conocido como **problema de clasificación binaria**.

Más formalmente, el aprendizaje automático supervisado [52] necesita de un **input**, el cual es el conjunto de datos de entrenamiento que es una secuencia finita $S = ((x_1, y_1), \ldots, (x_n, y_n))$ de pares pertenecientes a $X \times Y$. y_i es llamado el *label* correspondiente a x_i. El algoritmo tendrá un **output**, el cual es una función

$$h : X \rightarrow Y$$

llamada hipótesis, cuyo objetivo es predecir un $y \in Y$ correcto para algún $x \in X$ arbitrario, especialmente para aquellos que no están contenidos en el conjunto de datos de entrenamiento.

En nuestro trabajo, la tarea de clasificar si un mensaje de un estudiante es un forward o un backward looking act puede ser modelada como una tarea de aprendizaje automático supervisado ya que al anotar el conjunto de datos disponible con el label

de que el mensaje es un forward o backward looking act en el Capítulo 4, contamos con conjunto de datos donde para cada mensaje, tenemos una anotación y donde si $y = 1$, el mensaje es un forward looking act y si $y = 0$, el mensaje es un backward looking act. En nuestro caso, modelamos el problema como un problema de clasificación binaria donde cada $x \in X$ es un conjunto de características definidas que son extraídas de los mensajes e $Y = \{0, 1\}$. En la Sección 5.2 se analizará cómo se extraen las características de los mensajes.

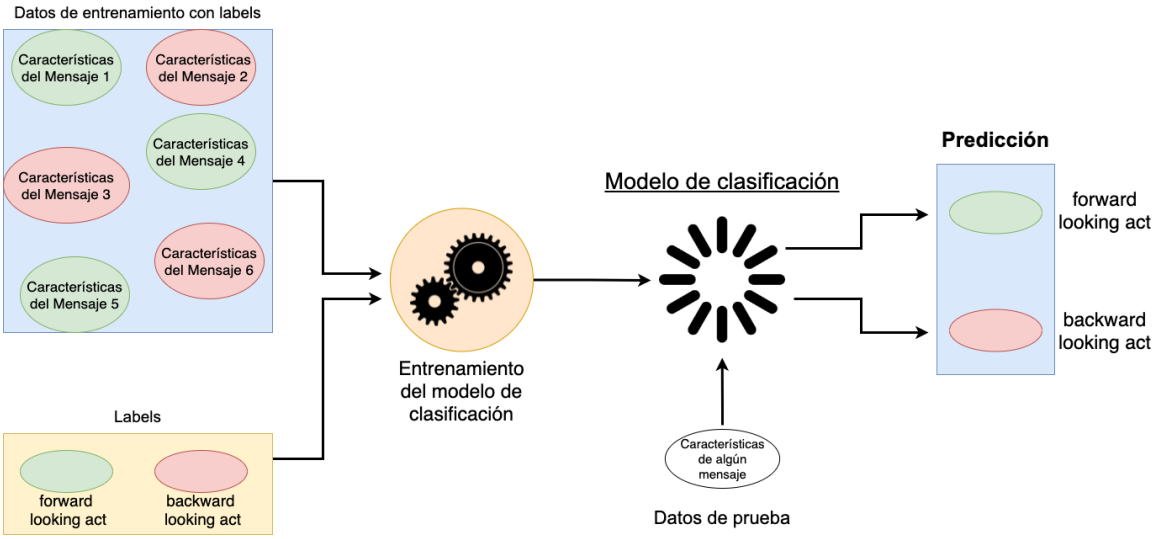


Figura 5.1: Diagrama del problema de aprendizaje automático supervisado

En la Figura 5.1 se muestra un diagrama que esquematiza el aprendizaje automático supervisado para resolver nuestro problema de clasificación binaria y los pasos que forman parte del proceso. Como primer paso, se cuenta con el conjunto de datos de entrenamiento. En nuestro problema, para cada mensaje del conjunto de datos se obtiene un conjunto de características extraídas del texto del mensaje y su respectivo label. En la Figura 5.1 los labels están representados con óvalos de colores, donde el óvalo verde corresponde a un forward looking act, y el óvalo rojo a un backward looking act.

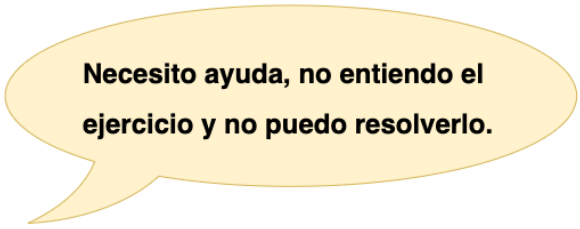
Una vez que se tiene el conjunto de entrenamiento, se elige un algoritmo de aprendizaje automático el cual luego de ser entrenado con los datos de entrenamiento, dará lugar al **modelo de clasificación**. El modelo de clasificación es capaz de tomar un conjunto de características de algún mensaje y puede, a partir del mismo, predecir si el conjunto de características dado corresponde con un mensaje que es un forward looking act o un backward looking act, basándose en lo aprendido del conjunto de entrenamiento. En la Figura 5.1 esto se puede ver cuando en "Predicción" se da como resultado un óvalo de color verde o rojo para las características de algún mensaje.

Para verificar que el modelo realice predicciones correctas, se utiliza un **conjunto de datos de prueba**, el cual consiste al igual que el conjunto de datos de entrenamiento, en un conjunto de características por cada mensaje, y sus respectivos labels. Con esta información, se le da al modelo las características de cada mensaje, y se compara la predicción dada por el modelo con el label que realmente le corresponde. En base a esta comparación, se pueden calcular métricas que determinan qué tan bien funciona el modelo de clasificación, las cuales se detallarán más adelante en la Sección 5.3.

En la sección a continuación se dará un ejemplo de cómo se extraen las características de un mensaje.

5.2. Características de los mensajes

Como vimos en la sección anterior, por cada mensaje del conjunto de datos es necesario obtener un conjunto de características representativas del mismo. Como nuestro conjunto de datos consiste en los strings de los mensajes, y los algoritmos de aprendizaje automático solamente pueden tomar valores numéricos como input, debemos seleccionar un conjunto de características que representen al mensaje y puedan ser representadas en un vector numérico. En la Figura 5.2 se puede ver un ejemplo de cómo se obtiene el vector numérico que representa las características del mensaje.



Características	a	ayuda	código	el	ejercicio	entiendo	ningún	necesito	no	porque
Cantidad de ocurrencias en el mensaje	0	1	0	1	1	1	0	0	2	0

Figura 5.2: Vector de características de un mensaje

En la primera fila de la tabla se presenta un ejemplo reducido a un vocabulario de 10 palabras a fines ilustrativos de características a usar como entrada: palabras elegidas y prefijadas con un determinado orden. Luego en la segunda fila, dado un mensaje, se forma el vector numérico correspondiente, donde cada valor corresponde con la cantidad de ocurrencias de la palabra en el mensaje. Por ejemplo, dado que en el mensaje no hay ninguna ocurrencia de *"a"*, su valor en el vector numérico es 0. También, dado que hay dos ocurrencias de *"no"*, su valor en el vector numérico es 2.

De esta manera, se obtiene un **vector de características** de un mensaje que puede ser utilizado como input en algoritmos de aprendizaje automático. Generalmente las palabras elegidas como características son palabras que potencialmente ocurren en un mensaje enviado en el foro de consultas: en el Capítulo 6 explicaremos qué características se usarán en los experimentos a realizar.

En la siguiente sección se detallan las principales métricas a utilizar en este trabajo, con el fin de poder evaluar qué tan bien funcionan los modelos.

5.3. Métricas de evaluación de modelos de aprendizaje automático

En esta sección se explica qué es una *matriz de confusión* y se presentan brevemente las métricas que se pueden obtener a partir de ella: exactitud (en inglés: *accuracy*), precisión (en inglés: *precision*), exhaustividad (en inglés: *recall*) y Valor-F1 (en inglés: *F1-Score*).

Una matriz de confusión es una matriz bidimensional en la cual cada columna representa el número de predicciones de cada clase, y cada fila representa las instancias en la clase real. De esta manera, permite la visualización del desempeño de un modelo de clasificación con respecto a un conjunto de datos de prueba.

		Predicción	
		Clase positiva	Clase negativa
Clase real	Clase positiva	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Clase negativa	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Tabla 5.1: Matriz de confusión

En la Tabla 5.1 se representa una matriz de confusión, en ella se puede observar que en la columna izquierda de las predicciones y en la primera fila de las clases reales se toman en cuenta los elementos de una clase llamada *positiva* y en la columna derecha derecha de las predicciones y la segunda fila de las clases reales se toman en cuenta

los elementos de una clase llamada *negativa*. Estos nombres surgen de, por ejemplo, si se tiene un conjunto de fotos y se desea identificar las fotos de gatos, entonces la clase positiva son las fotos de gatos, mientras que la clase negativa es la negación de esa clase positiva, es decir, las fotos que no son de gatos. En nuestro dominio en particular, tenemos dos clases: forward looking act y backward looking act, y si bien una clase es complemento de la otra, cada clase tiene una semántica en particular. En el ejemplo de las fotos de gatos, la clase negativa es el complemento de la positiva y no se conoce qué fotos son, en cambio en nuestro dominio, al conocer la semántica de ambas clases no podemos determinar cuál es la clase positiva y cuál es la clase negativa.

También podemos ver que el clasificador tiene dos maneras diferentes de fallar en su predicción: clasificando un elemento de la clase negativa como perteneciente a la clase positiva, lo que sería un falso positivo (FP) o clasificando un elemento de la clase positiva como perteneciente a la clase negativa, lo que sería un falso negativo (FN). Estos dos tipos de errores son conocidos como *Error de tipo 1* y *Error de tipo 2* respectivamente.

Como describimos en el Capítulo 4, luego de la anotación obtuvimos que de nuestro conjunto de datos el 64.9 % son forward looking acts, siendo la clase mayoritaria, y el 35.1 % son backward looking acts, siendo la clase minoritaria. Es por esto que el conjunto de datos no está balanceado (lo que sería aproximadamente 50 % y 50 %) y se trata de un problema de clasificación desbalanceada. Para los problemas de clasificación desbalanceada en general, se suele hacer que los elementos de la clase mayoritaria sean referidos como la clase negativa (conocidos también como clase 0) mientras que aquellos elementos de la clase minoritaria sean referidos como la clase positiva (conocidos también como clase 1). Es por esto que en nuestro trabajo se podría pretender utilizar los backward looking acts como la clase positiva, dado que son la clase minoritaria, y los forward looking acts como la clase negativa, dado que son la clase mayoritaria. En nuestro dominio cometer un error de tipo 1 es clasificar un forward looking act como backward looking act, lo que implicaría que el mensaje que necesita atención no sería clasificado como que no la necesita. Esto representa un problema posiblemente más grave que cometer un error de tipo 2, lo que implicaría clasificar un backward looking act como forward looking act, por lo que un mensaje que no necesita atención sería clasificado como que sí la necesita. Es por esto también que en nuestro dominio no está claro cuál es la clase positiva y cuál es la clase negativa ya que ambas clases, forward looking act y backward looking act son de nuestro interés, por lo que podríamos evaluar los modelos desde las dos perspectivas: una con forward looking acts como la clase positiva y otra con backward looking acts

como la clase positiva. Por lo tanto, en este capítulo y en el Capítulo 6 tendremos en cuenta el desempeño desde ambas perspectivas.

A partir de la matriz de confusión, se pueden obtener 4 métricas. Una de ellas es **accuracy**, la cual se puede calcular mediante la Ecuación 5.1 donde $Total = VP + FN + FP + VN$. La métrica accuracy describe la cantidad de predicciones correctas sobre todas las predicciones.

$$\text{Accuracy} = \frac{VP + VN}{Total} \tag{5.1}$$

Otra métrica es **precision**, la cual se calcula mediante la Ecuación 5.2. La métrica precision indica la fracción entre los casos que el clasificador predijo correctamente como positivos, sobre todos los casos predichos como positivos por el modelo.

$$\text{Precision} = \frac{VP}{VP + \mathbf{FP}} \tag{5.2}$$

Otra métrica es **recall**, que indica la fracción entre los casos que el clasificador predijo correctamente como positivos, sobre todos los casos positivos de los datos. Se puede calcular mediante la Ecuación 5.3.

$$\text{Recall} = \frac{VP}{VP + \mathbf{FN}} \tag{5.3}$$

La diferencia entre precision y recall está en el denominador de la fracción, como se puede ver resaltada en las fórmulas. Una buena precisión significa que el algoritmo devuelve más resultados correctos que incorrectos para la clase positiva, y un alto recall significa que el algoritmo clasifica como positivas la mayoría de las instancias positivas. La métrica **F1** permite combinar estas dos métricas en una. F1 se puede calcular mediante la Ecuación 5.4 o como la media armónica entre precision y el recall.

$$\text{F1} = \frac{VP}{VP + \frac{1}{2}(FP + FN)} \tag{5.4}$$

En la siguiente sección se analizarán más en detalle las métricas Accuracy y F1 aplicadas a nuestro dominio.

5.3.1. Accuracy vs. F1

En este trabajo, no nos enfocaremos en la métrica accuracy ya que si bien es de las más utilizadas, no distingue cuando el conjunto de datos disponible está desbalanceado.

		Predicción	
		Forward looking act	Backward looking act
Clase real	Forward looking act	5106	0
	Backward looking act	2753	1

Tabla 5.2: Matriz de confusión para un modelo de clasificación que siempre predice forward looking act excepto un caso donde predice backward looking act correctamente

En la Tabla 5.2 se presenta la matriz de confusión de un modelo que siempre predice forward looking acts salvo en una predicción, donde acierta la predicción de un backward looking act. Luego, en la Tabla 5.3 se calculan las métricas para cada una de las perspectivas mencionadas con anterioridad: una con forward looking act como clase positiva, y otra con backward looking act como clase positiva.

Clase positiva	Cantidad de elementos reales de la clase	Cantidad de elementos clasificados como pertenecientes a la clase	Accuracy	Precision	Recall	F1
Forward looking act	5107	7860	0.6498	0.65	1	**0.79**
Backward looking act	2754	1	0.6498	1	0.00036	**0.00073**

Tabla 5.3: Valores de accuracy, precision, recall y F1 para ambas perspectivas de clase positiva en el modelo que siempre predice forward looking act excepto un caso donde predice backward looking act correctamente.

En la primera fila de la Tabla 5.3 se toman los forward looking acts como la clase positiva y se calculan la cantidad de elementos que son forward looking act, la cantidad de elementos que el modelo clasifica como forward looking act y las métricas precision, recall y F1. Luego, en la segunda fila se toman los backward looking acts como la clase positiva y se calculan la cantidad de elementos que son backward looking act, la cantidad de elementos que el modelo clasifica como backward looking act y las métricas precision, recall y F1.

En ambas perspectivas, el accuracy tiene el mismo valor: 0.6498, mientras que el modelo en realidad sólo puede detectar un backward looking act de 2754 que hay en los datos. Además, se puede observar que el F1 obtenido para la perspectiva que toma forward looking act como clase positiva es 0.79, que es aún mayor que el accuracy, 0.6498.

En la segunda fila de la Tabla 5.3, al calcular las métricas con la perspectiva que toma los backward looking acts como clase positiva, el F1 da como resultado 0.00073, el cual refleja que el modelo no es capaz de identificar backward looking acts.

En la sección a continuación se detallan los modelos de aprendizaje automático

que se utilizarán en este trabajo.

5.4. Modelos de aprendizaje automático a utilizar

En esta sección se introducen tres algoritmos que se utilizarán en este trabajo: regresión logística, bosque aleatorio y red neuronal recurrente.

5.4.1. Modelo de regresión logística

En esta sección se detalla brevemente el modelo de regresión logística [33, 22] (en inglés: **logistic regression**). Como vimos en la Sección 5.1, para crear un modelo de aprendizaje automático, contamos con un vector de características $(x_1, x_2, ..., x_n)$ y el output y puede ser 1 o 0, ya que es un clasificador binario. En nuestro caso, el input será el vector de características de un mensaje obtenido de la manera ilustrada en la Sección 5.2 pero con un vocabulario de determinadas palabras dependiendo del experimento que se realice. Decimos que si el output del modelo es 1, el modelo predice que el mensaje que es un forward looking act, y si el output del modelo es 0, predice que el mensaje es un backward looking act; implicando que el mensaje requiere atención o no respectivamente. El algoritmo se llama "regresión" logística y el output termina siendo binario porque primero se realiza una regresión lineal y luego se lleva el valor obtenido (que puede ser cualquier número real) a un 1 o 0 utilizando una función, la cual veremos luego con más detalle. A continuación se detalla el funcionamiento del modelo.

El modelo intenta obtener una predicción $\hat{y}$ que debe ser igual a la etiqueta y de esa instancia, es decir $\hat{y} = y$. Para lograr esto, se necesita conocer la probabilidad $p(y = 1|x)$ que es la probabilidad de que la observación sea de la clase 1, mientras que $p(y = 0|x)$ es la probabilidad de que la observación sea de la clase 0. Regresión logística logra resolver este problema a través del aprendizaje de un vector de pesos w (en inglés: *weights*) y un sesgo b (en inglés: *bias*). Cada peso $w_i \in \mathbb{R}$ está asociado con una característica x_i. El peso w_i representa qué tan importante es esa característica en el input para la decisión de clasificación, y el peso puede ser mayor o menor que cero. El bias $b \in \mathbb{R}$ es sumado al input con los pesos ya aplicados. En el entrenamiento el modelo aprende los pesos w_i y el bias b, los cuales utilizará luego con el conjunto de datos de prueba para realizar predicciones de la siguiente manera: se multiplica cada x_i por su peso w_i y se suman los resultados para todo i hasta n. A esto se le suma el bias b y se obtiene un número z. El cálculo de z se puede ver en la Ecuación 5.5

$$z = \left(\sum_{i=1}^{n} w_i x_i\right) + b \tag{5.5}$$

También se puede calcular z utilizando el producto escalar obteniendo que $z = w \cdot x + b$. Pero la fórmula de z no necesariamente devuelve una probabilidad (un número entre 0 y 1). Para calcular una probabilidad, se pasa z por la función **sigmoide**, $\sigma(z)$. La función sigmoide también es llamada función logística y se calcula mediante la Ecuación 5.6.

$$y = \sigma(z) = \frac{1}{1 + \exp(-z)} \tag{5.6}$$

La función sigmoide tiene la ventaja de que su imagen es [0,1]. Ahora que se tiene un algoritmo que dado un x computa la probabilidad $p(y = 1|x)$, se debe determinar una heurística para tomar una decisión. La regresión logística toma esa decisión utilizando un *límite de decisión* (en inglés: *decision boundary*) el cual se muestra en la Ecuación 5.7 y es 0.5, ya que en la función sigmoide dada en la Ecuación 5.6 si $z > 0$ entonces $y > 0{,}5$ y si $z < 0$ entonces $y < 0{,}5$.

$$\hat{y} = \begin{cases} 1 & p(y = 1|x) > 0.5 \\ 0 & \text{caso contrario} \end{cases} \tag{5.7}$$

En la sección a continuación se detalla el modelo de bosque aleatorio.

5.4.2. Modelo de bosque aleatorio

En esta sección se detalla brevemente el modelo de bosque aleatorio [7] (en inglés *random forest*). Primero es necesario definir un **árbol de decisión** [6, 27]: un grafo árbol donde cada nodo no terminal está etiquetado con una característica del input. En nuestro caso, las características del input son palabras prefijadas, como se vió en la Sección 5.2. Cada nodo no terminal representa una condición que se verifica sobre la característica y en base a esta verificación, el árbol se parte en distintas ramas. En nuestro caso, la verificación que se realiza sobre la característica está relacionada con el valor que tiene en el vector de características, es decir, la cantidad de ocurrencias que hay de la característica en el mensaje. Cuando se llega a un nodo que ya no tiene particiones (una hoja del árbol), quiere decir que se llegó a un **nodo de decisión** y la clasificación queda determinada por la clase que el nodo de decisión indique. En nuestro caso, los nodos de decisión tendrán dos posibles clases: forward y backward looking act.

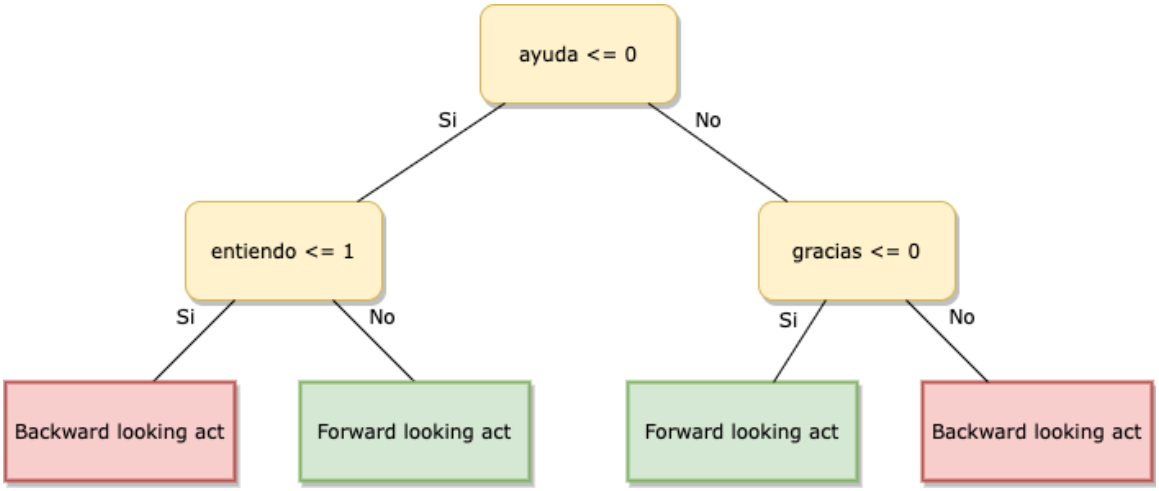


Figura 5.3: Ejemplo de árbol de decisión con 4 nodos de decisión y 3 características

En la Figura 5.3 se puede ver un ejemplo ilustrativo de árbol de decisión con tres características, donde el modelo toma el vector de características del mensaje, y a partir de él verifica las condiciones partiendo del nodo raíz del árbol, el cual verifica que la cantidad de veces que aparece la palabra *"ayuda"* en el mensaje sea menor o igual que 0. Así se va siguiendo un camino en el árbol a medida que se verifican las condiciones de los nodos no terminales (los cuales se ven de color amarillo) hasta llegar a algún nodo de decisión que determinará si el mensaje es un forward looking act (nodos de color verde) o un backward looking act (nodos de color rojo). Por ejemplo, si el mensaje es *"Necesito ayuda por favor."*, primero se evalúa el nodo raíz: como la palabra *"ayuda"* ocurre una vez y la palabra *"gracias"* no ocurre en el mensaje, el nodo de decisión al que se llega determina que el mensaje es un forward looking act. Luego, si el mensaje es *"Gracias, ya resolví el ejercicio."*, como la palabra *"ayuda"* no ocurre y la palabra *"entiendo"* tampoco ocurre en el mensaje, el nodo de decisión al que se llega determina que el mensaje es un backward looking act.

El modelo de árbol de decisión es interpretable ya que se pueden observar las condiciones que el modelo analiza sobre las características (lo que permite observar qué características son más importantes) y su flujo de decisión. Para lograr crear el modelo, los árboles de decisión utilizan el *coeficiente Gini* [11] para realizar particiones óptimas del conjunto de datos a partir de las características.

Un aspecto negativo de los árboles de decisión es que pueden *overfittear* los datos, es decir, es sensible a los datos específicos sobre los cuales se entrena: si el conjunto de datos de entrenamiento cambia, el árbol de decisión resultante puede ser diferente y por lo tanto, las predicciones pueden ser diferentes. Por esto es que bosque aleatorio combina múltiples árboles de decisión y obtiene un resultado único a partir de ellos,

produciendo un modelo de clasificación más robusto a cambios en los datos.

En la sección siguiente se detalla el modelo neuronal recurrente.

5.4.3. Modelo neuronal

En esta sección se detallan las redes neuronales que hacen uso de unidades Long short-term memory (LSTM).

Las redes neuronales son un subconjunto del aprendizaje automático y son la herramienta fundamental para los algoritmos de *aprendizaje profundo*. Una red neuronal recurrente (RNN, del inglés *recurrent neural network*) es un tipo de red neuronal que utiliza datos secuenciales o series de datos basados en tiempo. Este tipo de red permite crear modelos de aprendizaje automático que son comúnmente usados en problemas temporales u secuenciales, como traducción de textos, procesamiento de lenguaje natural, reconocimiento de voz [26], reconocimiento de texto escrito a mano [13], entre otros. A diferencia de las redes neuronales más simples (comúnmente llamadas *feed-forward*), las RNN se distinguen por su "memoria", ya que debido a su arquitectura, toman información de inputs anteriores para influenciar el input y output actual. Mientras que en las redes neuronales tradicionales, el input y el output son independientes uno del otro, el output de las RNN depende de elementos anteriores dentro de la secuencia de input. Una arquitectura RNN popular son las que poseen unidades LSTM, comúnmente llamadas redes LSTM [17], las cuales fueron originalmente introducidas como una solución al problema del desvanecimiento de gradiente [35]. En [17] trabajan el problema de las dependencias a largo plazo, esto es, si el estado previo que está influenciando la predicción actual no está en el pasado inmediato, el modelo de RNN quizás no pueda predecir precisamente el estado actual. Para solucionar este problema, las redes LSTM tienen celdas en las capas ocultas que poseen tres "puertas": una puerta de input, una puerta de output y una puerta de olvido. Estas puertas controlan el flujo de información que es necesario para predecir el output de la red neuronal.

La motivación de construir un modelo neuronal se da por el hecho de que, a diferencia de los modelos presentados anteriormente, una red LSTM respeta el orden de aparición de palabras en el mensaje y, a la hora de analizar una palabra de un mensaje, se cuenta con información de palabras que ocurrieron anteriormente, por lo cual es una posibilidad que esta información adicional implique una mejora a la hora de la clasificación. Una diferencia importante es que a la RNN se le puede dar como input el texto plano del mensaje, lo que implica que no es necesario realizar construir un vector de características a partir de un mensaje para dar como input. El modelo a utilizar se puede observar en la Figura 5.4: primero, una capa de Embedding [29] de e

dimensiones es alimentada con las palabras del mensaje, las cuales se obtienen a partir de un proceso llamado *tokenización* que se verá más en detalle en el Capítulo 6. Esta capa de embedding luego será la entrada de una capa de 100 unidades LSTM. Por último, esta capa de LSTM estará conectada con una capa de activación sigmoide, la cual permite que el resultado de la red sea un valor 1 o 0, lo cual indicará si el mensaje es un forward o backward looking act respectivamente. Esta arquitectura de red es comúnmente conocida como *many to one*, (en español: muchos a una), ya que son numerosos inputs sobre la línea de temporalidad, para obtener un único resultado.

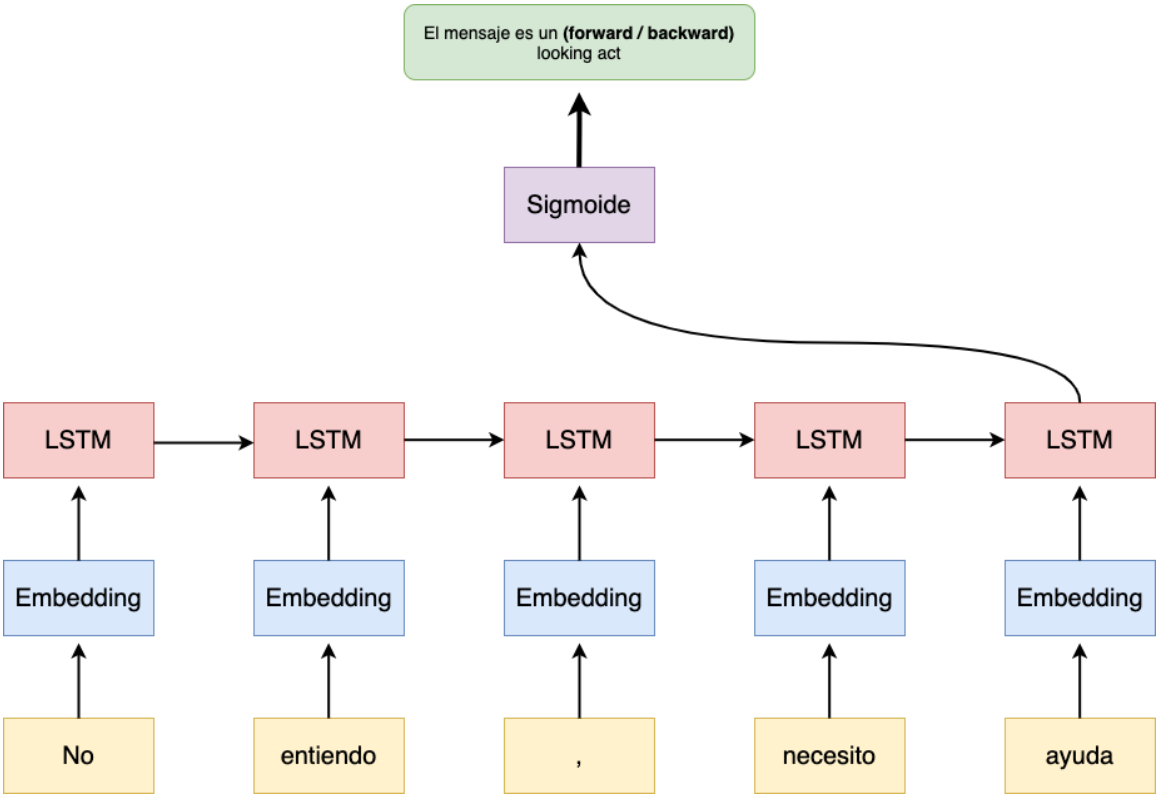


Figura 5.4: Arquitectura de la red LSTM

En la sección a continuación se detalla la técnica de validación cruzada, una técnica comúnmente utilizada para evaluar modelos de clasificación.

5.5. Validación cruzada

En esta sección se detalla la técnica de validación cruzada, la cual se utiliza para garantizar la independencia de resultados de un modelo de clasificación con respecto a los datos de entrenamiento y prueba seleccionados.

Durante el entrenamiento de modelos de aprendizaje automático, el cálculo de las

métricas vistas en la Sección 5.3 frecuentemente no permiten llegar a conclusiones realistas sobre el comportamiento del modelo ante nuevas observaciones, ya que estas métricas dependen de los conjuntos de datos de entrenamiento y prueba, es decir, las métricas varían considerablemente de acuerdo a los elementos que conforman el conjunto de datos de entrenamiento y prueba.

Para evitar esto, es común utilizar la técnica de **validación cruzada** (en inglés: *cross validation*). Uno de los tipos de cross validation es conocido como *K-fold cross validation* que consiste en dividir el conjunto de datos anotado en K subconjuntos. Uno de los subconjuntos se utiliza como datos de prueba y los K-1 subconjuntos restantes como datos de entrenamiento. Este proceso es repetido durante K iteraciones, donde cada uno de los posibles subconjuntos es utilizado como datos de prueba. Por último, se toma la media de los resultados de cada iteración para obtener un resultado único.

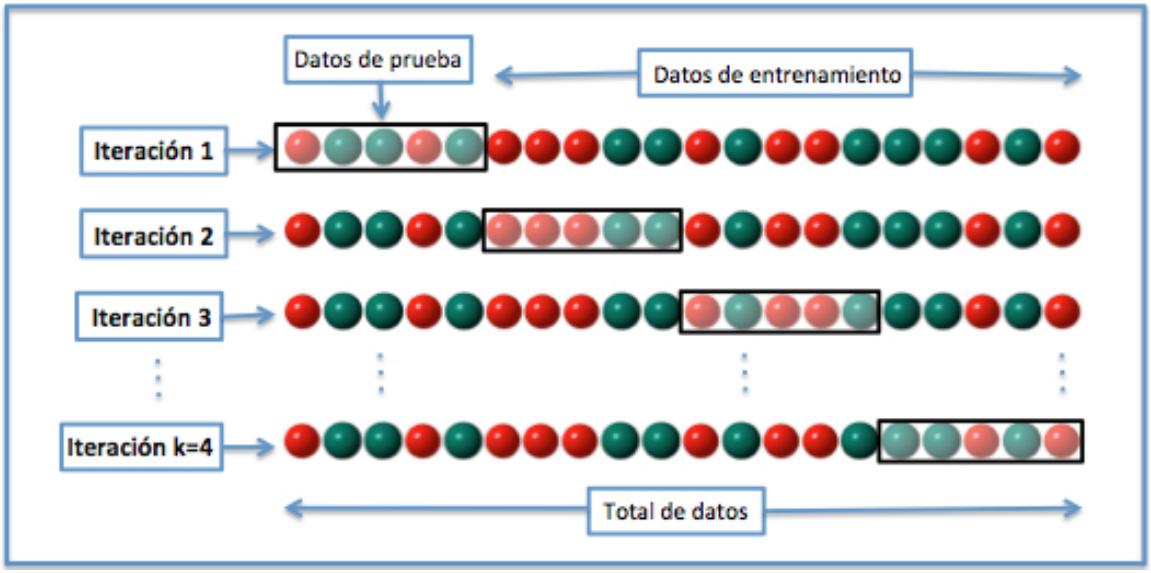


Figura 5.5: K-fold cross validation con K = 4

En la Figura 5.5 se puede ver un ejemplo gráfico del funcionamiento de K-fold cross validation con K=4. A pesar de que K-fold cross validation es una técnica para evitar la dependencia de las métricas de los conjuntos de datos de entrenamiento y prueba, el resultado de cross validation depende de cómo se particionan los conjuntos para prueba y entrenamiento (los *folds*). Para evitar esto, se utiliza cross validation repetidas veces, lo cual es llamado **Repeated K-fold cross validation**. Con esta técnica, se ejecuta k-fold cross validation n veces, donde en cada una de las n ejecuciones, el conjunto de datos se particiona de diferentes maneras, obteniendo folds diferentes. Finalmente se computa la media de los resultados de las n ejecuciones y se obtiene el

resultado único. Estas técnicas pueden ser costosas computacionalmente porque hay que reentrenar el modelo para cada nuevo conjunto entrenamiento. Tampoco aseguran que el desempeño que predigan se mantenga en el momento de producción si los nuevos datos a clasificar difieren de aquellos presentes en el total de datos que se usó para entrenamiento y prueba.

5.6. Relación con otros capítulos

En este capítulo se describió cómo la tarea de determinar si un mensaje enviado en el foro de consultas de Mumuki por un estudiante requiere la atención de un mentor o no, de manera automática, puede ser modelada como un problema de clasificación binaria de aprendizaje supervisado. También se ilustró la manera de obtener el vector de características de un mensaje, lo cual será útil en el siguiente capítulo para poder explicar cómo entrenamos los modelos de regresión logística y bosque aleatorio. Además, se analizaron las métricas a utilizar durante los experimentos para poder medir el desempeño de los modelos de clasificación. Luego, se presentaron los distintos tipos de modelos de aprendizaje automático que serán utilizados. Por último, se presentó la técnica de validación cruzada para mejorar la independencia de los resultados con respecto a la partición que se realice del conjunto de datos en prueba y entrenamiento.

Los resultados obtenidos al aplicar los modelos presentados en este capítulo a nuestro problema se presentarán en el Capítulo 6. Luego, en el Capítulo 7 se presentarán las conclusiones de este trabajo.

Capítulo 6

Experimentos y resultados

En este capítulo se presentan los resultados de los distintos experimentos que utilizan aprendizaje automático para modelar el problema de clasificar si un mensaje es un forward looking act o un backward looking act. En la Sección 6.1 se reportará el desempeño de un modelo básico de clasificación para tener como base a la hora de hacer comparaciones entre modelos. Al modelo base se lo llamará por su nombre en inglés: *baseline*. En la Sección 6.2 se presentan los resultados de los modelos de aprendizaje automático de regresión logística y bosque aleatorio con bolsa de palabras. Luego, en la Sección 6.3 se presentan los resultados de un modelo neuronal secuencial.

6.1. Baseline

En trabajos donde se aplica aprendizaje automático se suele establecer un modelo básico de predicción para tener como base y compararlo con distintos modelos que se construyan posteriormente. A estos modelos se los conoce como **baseline** (*base de referencia* en español). En nuestro trabajo, utilizaremos como baseline un modelo que realiza predicciones utilizando *muestreo aleatorio estratificado*. Muestreo aleatorio estratificado es un método de muestreo que implica particionar la población que se está estudiando en grupos más reducidos llamados *"estratos"*. Los grupos o estratos se organizan basándose en características o atributos similares que tengan los elementos del estrato. Luego, la muestra aleatoria que se toma depende de la proporción que los estratos representan de la población. En nuestro caso, los estratos son los forward y los backward looking acts. En nuestro conjunto de datos tenemos 5107 elementos de la clase forward looking act y 2754 elementos de la clase backward looking act, representando el 64.9 % y 35.1 % respectivamente, es por esto que nuestro modelo baseline que realiza un muestreo aleatorio estratificado va a predecir, independientemente del input que se le de, un forward looking act con una probabilidad $p_{forward} = 0.649$ y

un backward looking act con probabilidad $p_{backward} = 0.351$.

		Predicción	
		Forward looking act	Backward looking act
Clase real	Forward looking act	*#Forward looking acts* * $p_{forward}$ 5107 * 0.649 = **3314**	*#Forward looking acts* * $p_{backward}$ 5107 * 0.351 = **1793**
	Backward looking act	*#Backward looking acts* * $p_{forward}$ 2754 * 0.649 = **1787**	*#Backward looking acts* * $p_{backward}$ 2754 * 0.351 = **967**

Tabla 6.1: Matriz de confusión para el modelo baseline considerando la esperanza

En la Tabla 6.1 se presenta la matriz de confusión para el modelo baseline. Dado que el modelo realiza predicciones de manera estocástica, la predicción del modelo para un mismo input puede diferir cada vez que se realice una nueva predicción, por lo que para obtener la matriz de confusión se toma la esperanza de predicciones. Por ejemplo, para calcular la esperanza de cantidad de elementos que son forward looking act y el modelo predice como tal, se pueden pensar esas predicciones como una variable aleatoria $X \sim Bin(n,\ p)$[1] tal que X mide la cantidad de predicciones forward looking act, con $E[X] = n\,p$, con $n = 5107$ y $p = 0.649$ ya que el modelo hace 5107 predicciones independientes donde la clase real del elemento es forward looking act y la probabilidad de que el modelo prediga que es forward looking act es 0.649. Por lo tanto, la esperanza de predicciones de forward looking acts como tales es 5107 * 0.649 = 3314. De manera similar se calculan los demás casos.

Forward looking act		
Precision	Recall	F1
0.649	0.649	0.649

Backward looking act		
Precision	Recall	F1
0.351	0.351	0.351

Tabla 6.2: Resultados del modelo baseline

A partir de la matriz de confusión de la Tabla 6.1 se calculan las métricas para ambas perspectivas de clase positiva, las cuales se presentan en la Tabla 6.2. El encabezado de la tabla indica qué clase se considera la clase positiva. En la tabla se observa que si se toman los forward looking acts como clase positiva, el modelo llega a obtener un F1 de 0.649. Mientras tanto, si se toman los backward looking acts como clase positiva, el modelo llega a obtener un F1 de 0.351. Estos resultados en el baseline permiten poder tener una base de referencia y realizar comparaciones con modelos de experimentos posteriores.

En la próxima sección se realizan experimentos utilizando bolsa de palabras.

[1] https://es.wikipedia.org/wiki/Distribución_binomial

6.2. Bolsa de palabras

En esta sección se realizan experimentos utilizando el modelo *"bolsa de palabras"*, el cual es una representación simplificada que se utiliza en el procesamiento del lenguaje natural. En este modelo, un texto (en nuestro caso, un mensaje) se representa como una bolsa de sus palabras, sin tener en cuenta el orden de las palabras pero manteniendo su frecuencia de ocurrencia en el texto. Este modelo se usa comúnmente en modelos de clasificación de documentos donde la ocurrencia de cada palabra se usa como una característica para entrenar al clasificador. En esta sección se utilizan modelos de aprendizaje automático a los cuales se les dará como input una bolsa de palabras que se representa mediante el vector de características de un mensaje, el cual vimos un ejemplo de cómo se construye en el Capítulo 5. En la Sección 6.2.1 se utilizan los modelos de regresión logística y bosque aleatorio donde para la obtención del vector de características de un mensaje se utilizan como características palabras representativas de mensajes forward y backward looking acts que fueron previamente analizadas en el Capítulo 4. Luego, en la Sección 6.2.2 también se utilizan los modelos de regresión logística y bosque aleatorio, pero utilizando como características todas las palabras que ocurren en todos los mensajes del conjunto de datos, lo cual se conoce como *vocabulario*.

6.2.1. Utilizando un conjunto de palabras determinado

Como primer experimento, se utilizaron como características las estudiadas en el Capítulo 4, es decir, 8 n-gramas con n = 1, 2 y 3 representativos de mensajes que son forward looking acts y 19 n-gramas con n = 1, 2 y 3 representativos de mensajes que son backward looking acts. También se da como input la longitud del mensaje en caracteres. Se entrenaron modelos de regresión logística y bosque aleatorio utilizando la librería scikit-learn [36] en Python. El entrenamiento y partición del conjunto de datos se realizó utilizando *Repeated K-fold cross validation* con $k = 10$ y 3 repeticiones, explicada en el Capítulo 5. Debido a que la característica de "longitud de mensaje" tiene valores en rangos distintos al de las demás características, se normaliza el conjunto de datos utilizando *StandardScaler* de scikit-learn, el cual utiliza como estrategia restar la media y se dividir por la desviación estándar en cada característica independientemente.

En la Tabla 6.3 se muestran los resultados obtenidos para los modelos de regresión logística, bosque aleatorio con 1 árbol de decisión, 10 árboles de decisión, 100 árboles de decisión y 1000 árboles de decisión para las perspectivas que toman forward looking act y backward looking act como clase positiva.

	Forward looking act			Backward looking act		
	Precision	Recall	F1	Precision	Recall	F1
Regresión logística	0.794	0.912	0.848	0.774	0.560	0.650
Bosque aleatorio (1)	0.826	0.846	0.836	0.702	0.670	0.685
Bosque aleatorio (10)	0.828	0.863	0.845	0.724	0.668	0.694
Bosque aleatorio (100)	0.827	0.867	0.846	0.730	0.664	0.695
Bosque aleatorio (1000)	0.827	0.868	0.847	0.731	0.664	0.695

Tabla 6.3: Resultados de utilizar las características estudiadas en el Capítulo 4

Este experimento logra una mejora en la métrica F1 del 30 % en forward looking acts y del 98 % en backward looking acts con respecto al modelo baseline: mientras que en los forward looking acts se lograba el F1 más alto con valor 0.649, en este experimento se logra un F1 de 0.847; y mientras que en los backward looking acts se lograba el F1 más alto con valor 0.351, en este experimento se logra un F1 de 0.695.

En la sección que sigue se realiza un segundo experimento utilizando todo el vocabulario de los mensajes del conjunto de datos como características.

6.2.2. Utilizando todo el vocabulario

Como segundo experimento, también se entrenaron modelos de regresión logística y bosque aleatorio, utilizando *Repeated K-fold cross validation* con $k = 10$ y 3 repeticiones. En este experimento, se realiza como primer paso un preprocesamiento del conjunto de datos: se segmenta el texto de los mensajes en *tokens* (procedimiento llamado *tokenización*), que son una secuencia de caracteres utilizados como la unidad de análisis de texto. Es común pensar al token como una palabra del lenguaje pero además los tokens pueden ser un símbolo de puntuación u otro tipo de símbolo. Un token puede ser una palabra como *"hola"*, un número, un solo caracter o un símbolo como *"?"*, o *"{ "*. En este experimento también se da como input la longitud en caracteres del mensaje y además, en vez de utilizarse ciertos n-gramas definidos como características de un mensaje para formar el vector de características, se utilizan todos los tokens que ocurren en todos los mensajes del conjunto de datos, a lo cual llamamos **vocabulario**. Luego de la tokenización, se convierten todos los caracteres de los tokens a minúscula, lo cual reduce el tamaño del vocabulario, de esta manera, por ejemplo, los tokens *"Ayuda"* y *"ayuda"* pasan a ser un mismo token en el vocabulario. En el área de extracción de información se utilizan las técnicas de *lematización* o *stemming* sobre los tokens. La lematización es un proceso lingüístico que consiste en que dada una palabra, se halla su "lema". El lema es la forma que por convenio se acepta como representante de todas las formas de una misma palabra.

Por ejemplo, *"comienza"*, *"comenzarán"* y *"comienzo"* se lematizan en *"comenzar"*. Stemming consiste en el acortamiento de las palabras para llevarlas una base común, por ejemplo: *"comienza"* se reduce en *"comienz"*, *"comenzarán"* en *"comenz"*. En este trabajo, no aplicaremos stemming dado que no funciona bien para español y tampoco lematización dado que las declinaciones de palabras pueden ser relevantes para indicar si un mensaje es forward o backward. Por ejemplo, los forwards prototípicamente hablan del futuro mientras que los backwards hablan del pasado. Otra técnica que se suele utilizar es omitir un conjunto de tokens del vocabulario, este conjunto se lo llama *stopwords*, y suelen ser tokens que ocurren frecuentemente, como artículos y preposiciones. En nuestro caso la eliminación de stopwords fue contraproducente en los resultados, por lo que se descartó su uso. Esto puede deberse a que el modelo de clasificación es capaz de obtener información valiosa de stopwords posibles como artículos y preposiciones. Con este procesamiento del conjunto de datos, se obtiene un vocabulario de 8128 tokens. Los mejores resultados se dieron al intentar reducir el vocabulario, por lo que en el experimento final se consideran como parte del vocabulario tokens que ocurran en por lo menos 5 mensajes del conjunto de datos, lo cual reduce el vocabulario a 1645 tokens. Por cuestiones de reproducibilidad mencionamos que para la tokenización se utilizó la librería *nltk*, para la obtención del vocabulario se utilizó *CountVectorizer* de scikit-learn y para la normalización se utilizó StandardScaler de scikit-learn.

	Forward looking act			**Backward looking act**		
	Precision	Recall	F1	Precision	Recall	F1
Regresión logística	0.931	0.939	0.935	0.886	0.871	0.878
Bosque aleatorio (1)	0.889	0.895	0.892	0.804	0.793	0.798
Bosque aleatorio (10)	0.943	0.945	0.944	0.898	0.894	0.896
Bosque aleatorio (100)	0.942	0.966	0.954	0.933	0.891	0.911
Bosque aleatorio (1000)	0.942	0.968	0.955	0.937	0.890	0.913

Tabla 6.4: Resultados de usar todo el vocabulario como características

En la Tabla 6.4 se muestran los resultados obtenidos para los modelos regresión logística, bosque aleatorio con 1 árbol de decisión, 10 árboles de decisión, 100 árboles de decisión y 1000 árboles de decisión para las perspectivas que toman forward looking act y backward looking act como clase positiva.

La utilización del vocabulario como características logra una mejora en la métrica F1 del 12 % en forward looking acts y del 31 % en backward looking acts con respecto al experimento de la Sección 6.2.1: mientras que en los forward looking acts se lograba el F1 más alto con valor 0.847, en este experimento se logra un F1 de 0.955; y mientras

que en los backward looking acts se lograba el F1 más alto con valor 0.695, en este experimento se logra un F1 de 0.913.

Otro aspecto a tener en cuenta es el valor alto de recall que se logra en los forward looking acts: 0.968 en el modelo de bosque aleatorio con 1000 árboles de decisión. Esto indica que el modelo logra clasificar como forward looking act el 96.8 % de los forward looking acts, minimizando así los casos donde se clasifica un forward looking act como backward looking act. Esto es importante para nuestra tarea, como se mencionó en el Capítulo 5.

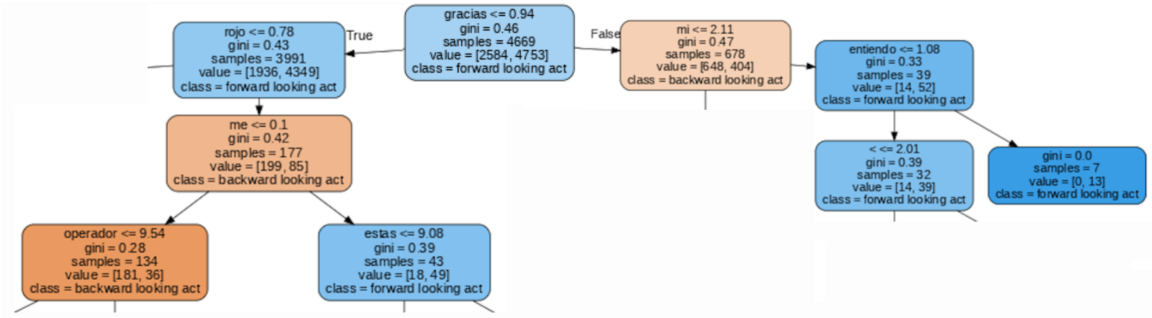


Figura 6.1: Parte de uno de los árboles de decisión de BA100

En la Figura 6.1 se puede ver la parte superior de uno de los 100 árboles de decisión del modelo de bosque aleatorio con 100 árboles de decisión **(BA100)**, donde se observa que el nodo raíz realiza la primera partición del conjunto de datos dependiendo de la cantidad de veces que ocurre la palabra *"gracias"* en el mensaje. Un ejemplo de decisión del clasificador BA100 es que si el mensaje contiene la palabra *gracias* más de 0.94 veces, la palabra *mi* más de 2.11 veces y la palabra *entiendo* más de 1.08 veces (donde todos estos valores están normalizados), entonces el mensaje es un forward looking act. También, se puede observar que la palabra palabra *"mi"* es utilizada por el árbol de decisión para realizar una clasificación: suele ocurrir en los backward looking acts tales como *"Hola, a mi me funcionó así: [...]"* o *"Ya me dí cuenta de mi error [...]"* pero no ocurre tan frecuentemente como en forward looking acts, que generalmente son de la forma *"No entiendo que tengo mal en mi código [...]"* o *"Me dice que mi solución esta mal porque [...]"*. Esto puede suceder ya que es más común hablar de uno mismo cuando está realizando un forward looking act, ya que está "pidiendo" o "esperando" algo del oyente, mientras que cuando se realiza un backward looking act se está "respondiendo" o "mirando atrás" algún mensaje previo de otra persona.

En un análisis manual no sistemático de las predicciones los modelos de clasificación parecen tender a identificar mensajes que contienen código como backward looking

acts, como se mencionó en el Capítulo 4 al describir la característica de longitud de mensaje.

Mensaje	Predicción
Hola! Buen día, tendrías que escribirlo de otra manera	Forward looking act
Hola! Buen día, tendrías que escribirlo de otra manera: ***program*** **{}**	Backward looking act

Tabla 6.5: Ejemplo de clasificación de BA100

Un ejemplo de este comportamiento se observa en la Tabla 6.5, si al mensaje *Hola! Buen día, tendrías que escribirlo de otra manera* (el cual es un backward looking act dado que es un mensaje que no genera o implica ninguna obligación en el oyente) le agregamos al final *"program{}"*, que es una palabra clave frecuente en el lenguaje de programación Gobstones, el cual es uno de los lenguajes que se utiliza en el programa de estudio de Mumuki, el modelo de clasificación BA100 pasa de clasificarlo como forward looking act a backward looking act. Es decir, BA100 reconoce que el mensaje es un backward looking act por el hecho de agregar un fragmento de código al mensaje, y no detecta originalmente que el mensaje original es un backward looking act. Esto da indicios de que los modelos pueden estar utilizando palabras propias de código de Gobstones para predecir backward looking acts.

En la sección que sigue se realiza un tercer experimento utilizando el modelo neuronal recurrente.

6.3. Modelo neuronal

En esta sección se presentan los resultados obtenidos con el modelo neuronal detallado en el Capítulo 5.

Para la construcción del modelo se utilizó la librería Keras de python. Este experimento, al igual que en los anteriores, se lleva a cabo utilizando *Repeated K-fold cross validation* con k=10 y 3 repeticiones. La tokenización de los mensajes se realizó de la misma manera que en el experimento de la Sección 6.2.2. Además, este experimento se realizó para distintas dimensiones de embeddings e: 64, 128, 256 y 512. Los resultados para las perspectivas que toman forward looking act y backward looking act como clase positiva se pueden observar en la Tabla 6.6.

En el modelo neuronal, la clasificación de backward looking acts parece estar menos condicionada por la presencia de código en el mensaje que en los modelos de experimentos anteriores, lo cual puede ser el motivo del aumento del recall de la perspectiva que toma backward looking acts como clase positiva con respecto al experimento de la Sección 6.2.2, donde el recall más alto fue de 0.890, mientras que en este experi-

	Forward looking act			Backward looking act		
	Precision	Recall	F1	Precision	Recall	F1
LSTM (Embeddings 64)	0.954	0.950	0.952	0.909	0.914	0.911
LSTM (Embeddings 128)	0.957	0.951	0.954	0.911	0.921	0.915
LSTM (Embeddings 256)	0.958	0.953	0.955	0.914	0.922	0.918
LSTM (Embeddings 512)	0.956	0.952	0.954	0.912	0.918	0.915

Tabla 6.6: Resultados del modelo neuronal

mento LSTM con embeddings de 256 dimensiones logra el recall más alto: 0.918. Esta diferencia de recall representa una disminución del error del 25.4 %.

En la Tabla 6.7 se puede observar la diferencia de clasificación sobre el mensaje observado a modo de ejemplo en la Sección 6.2.2, en la Tabla 6.5, entre el modelo LSTM de embeddings de 256 dimensiones y BA100. El modelo neuronal parece tener una mayor capacidad de evitar basarse en símbolos de fragmentos de código para clasificar backward looking acts. Cuando no está presente "*program*{}" en el texto, BA100 del experimento de la Sección 6.2.2 lo identifica como un forward looking act y luego, cuando se lo incluye, sí lo identifica como un backward looking act (como se vió en la Tabla 6.5), mientras que el modelo neuronal clasifica como backward looking act ambos textos, con lo cual la predicción es más robusta a esta perturbación.

	Modelo	
Mensaje	**LSTM (Embeddings 256)**	**BA100**
Hola! Buen día, tendrías que escribirlo de otra manera	Backward looking act	Forward looking act
Hola! Buen día, tendrías que escribirlo de otra manera: ***program{}***	Backward looking act	Backward looking act

Tabla 6.7: Diferencia de clasificación entre el modelo neuronal y BA100 sobre un mensaje

La mejor precision de este experimento para la perspectiva que toma los backward looking acts como clase positiva es un 2.6 % más baja que en el experimento la Sección 6.2.2 (0.937 contra 0.912), por lo que presenta un aumento del error de tipo 1, ya que al tener menor precision, hay más casos en los que predice forward looking acts como backward looking acts (más casos falsos positivos).

Además en este experimento, en la perspectiva que toma los forward looking acts como clase positiva, el mejor recall es un 1.5 % más bajo que en el experimento la Sección 6.2.2 (0.968 contra 0.953) y la mejor precision es un 1.5 % más alta (0.943 contra 0.958). Es por esto que la diferencia de mejores puntajes F1 entre ambos experimentos para esta perspectiva es de 0.001 puntos.

6.4. Relación con otros capítulos

En este capítulo se presentaron los resultados de los experimentos realizados utilizando modelos de aprendizaje automático presentados en el Capítulo 5. Los dos primeros experimentos fueron utilizando bolsa de palabras con los algoritmos de regresión logística y bosque aleatorio, uno con el vocabulario completo y otro con un vocabulario pequeño relevante que se describe en el Capítulo 4. Luego, se realizó un experimento utilizando una red neuronal recurrente LSTM. En el Capítulo 7 se dará lugar a la conclusión, al trabajo futuro a seguir y a las consideraciones éticas del trabajo.

Capítulo 7

Conclusiones y trabajo futuro

7.1. Conclusiones

En este trabajo se presentó un conjunto de datos sobre el cual no se había trabajado anteriormente, se lo limpió teniendo especial cuidado con la anonimización de estudiantes, se propuso y realizó un proceso de anotación del mismo. Se crearon modelos de aprendizaje automático que predicen si un mensaje escrito por un estudiante en el foro de consultas del sistema de enseñanza online de programación Mumuki es un forward looking act o un backward looking act de acuerdo a la teoría lingüística de actos de habla. En la teoría de actos de habla un forward looking act es un acto de habla, es decir, algo que dice un participante de una conversación, que requiere una respuesta u otro tipo de reacción del interlocutor (por ejemplo, una pregunta, una orden, etc). Un forward looking act impone una obligación en el interlocutor que, si no se cumple disminuye la sensación de colaboración en la conversación. Un backward looking act es un acto de habla que satisface una obligación conversacional y que no genera otra obligación (por ejemplo, una respuesta a una pregunta como "sí", un agradecimiento, un saludos de despedida). El conjunto de datos sobre el que se trabajó fue provisto por la empresa Ikumi, creadora del software Mumuki. Consiste en los datos de las consultas realizadas en el foro y los mensajes enviados en ellas. Del conjunto de datos se pudo observar que en el dominio de las plataformas de aprendizaje de programación en línea, y en particular en Mumuki, sucede que los mensajes backward looking acts de los estudiantes tienden a ser más extensos que los mensajes forward looking acts. En otros dominios de diálogo, por ejemplo, en sistemas de mensajería instantánea para hacer consultas sobre compras, frecuentemente, los forward looking acts son más largos que los backward looking acts. En nuestro dominio, la longitud extensa de los backwards looking acts se debe generalmente a la presencia de código en los mensajes backward looking act. Muchas veces a la hora de

responder consultas, los participantes del foro incluyen código en sus mensajes para orientar al estudiante que realiza la consulta. Las tareas realizadas pueden resumirse en los siguientes tres pasos. Primero, para realizar la anotación del conjunto de datos, se diseñó una herramienta online que permitió a tres anotadores clasificar los mensajes en forward o backward looking acts, lo cual implica, en la teoría lingüística de actos de habla, que requieren o no atención de un mentor respectivamente. Segundo, para la construcción de modelos de aprendizaje automático, primero se utilizaron algoritmos de regresión logística y bosque aleatorio utilizando como input bolsas de palabras. Para la definición de las bolsas de palabras, se experimentó primero con un conjunto definido de palabras que caracterizan a los forward y backward looking acts, y luego se experimentó con utilizar todas las palabras del conjunto de entrenamiento del conjunto de datos. Por último, se construyó una red neuronal recurrente la cual no necesita del diseño de características específicas y utiliza como input el texto plano del mensaje. En los experimentos de modelos mostramos un ejemplo de mensaje en nuestro dominio y cómo un modelo de bosque aleatorio se basa en la presencia de una palabra reservada para clasificarlo como backward looking act, lo cual no ocurre para el modelo neuronal. El modelo neuronal parece ser más robusto a pequeños cambios en el input y por lo tanto podría ser más difícil engañar al sistema (como se explica en el Capítulo 2) .

Se construyeron modelos de aprendizaje automático que detectan si un mensaje de un estudiante enviado en el foro de consultas es un forward o backward looking act, dependiendo del mensaje escrito, con valores F1 por encima de 0.9. Además, son modelos que están listos para ser integrados al sistema online Mumuki ya que los tiempos de clasificación de los modelos son cortos, lo que permite que sean utilizados en una web sin delay que afecte a la experiencia del usuario. El hecho de poder contar con un modelo que predice si el mensaje de un estudiante es un forward o backward looking act podría permitir al docente saber qué mensajes priorizar proporcionado una guía para identificar, dentro de los numerosos mensajes enviados por estudiantes en el foro, aquellos que necesitan respuesta de un docente. De esta manera la tarea docente puede enfocarse en priorizar su atención a estudiantes que requieren respuesta y así brindar respuestas con menos tiempo de espera. Esta dinamización de la interacción docente-estudiante (la cual ya de por sí se ve debilitada debido al contexto en línea mismo y la masividad de estudiantes) podría fomentar la participación de los estudiantes en los foros al notar que reciben respuestas más rápidamente. Además, al responder a tiempo las dudas de los estudiantes, estos podrían sentir más la presencia docente y evitar así posibles frustraciones o abandono por falta de atención a tiempo.

Hay distintas líneas en las que se puede seguir trabajando sobre la tarea y la

experiencia en los foros de consultas de Mumuki, lo cual da lugar a la siguiente sección, donde se mencionan posibles trabajos a seguir.

7.2. Trabajo futuro

Luego de haber construido los modelos que pueden ser puestos en funcionamiento en Mumuki, surgen posibles mejoras tanto para los modelos como para la experiencia de usuario en el foro de consultas.

- **Optimizar los hiper parámetros de la red neuronal exhaustivamente**: A pesar de haber conseguido un buen desempeño con la red neuronal, se puede realizar una búsqueda exhaustiva de los hiper parámetros de la misma tal que logren un resultado óptimo para las entradas consideradas.
- **Probar otra estructura de red neuronal**: El desempeño de la red neuronal puede seguir mejorando a través de experimentos de distintas arquitecturas de red neuronal, por ejemplo, se podría probar el desempeño de una red neuronal LSTM bidireccional, la cual además de proveer contexto del pasado, puede dar información acerca del futuro.
- **Mejorar la representación del texto de los mensajes**: La detección de código presente en el texto plano de los mensajes y su reemplazo por un token representativo como [CODIGO] podría significar una mejora en los resultados ya que todos los fragmentos de códigos serían vistos como un único token por los modelos, lo que limpia y diferencia la sintaxis de lenguajes de programación del lenguaje natural humano en los mensajes, lo que podría permitir un mejor análisis semántico. Otra mejora podría ser experimentar en la red neuronal con word embeddings pre-entrenados para el idioma español, aunque esa área no está tan desarrollada como sí lo está para el idioma inglés.
- **Enriquecer el input de los modelos**: Se puede enriquecer el input de los modelos con más información relacionada a la conversación en la cual se escribe el mensaje analizado. Por ejemplo, se puede dar un indicador de si el mensaje previo es de un mentor, o proveer más información acerca del tipo de estudiante que escribe el mensaje como por ejemplo su antigüedad en Mumuki, cantidad de ejercicios resueltos o cantidad de mensajes enviados en el foro.
- **Obtener un conjunto de datos más grande**: Un posible experimento sería trabajar con un conjunto de datos que tenga más mensajes, lo cual podría significar una mejora en los resultados ya que los modelos podrían lograr mayor

capacidad de generalización debido a una mayor cantidad de datos, evitando así concentración en unos pocos datos (overfitting) y mejorando el desempeño al ser puestos a prueba ante datos nunca antes vistos.

- **Clasificar el mensaje del estudiante mientras este lo redacta**: Una posible mejora de experiencia de usuario sería realizar una clasificación cualitativa del mensaje mientras es escrito por el estudiante y que no solo clasifique la necesidad de atención de un mentor, sino que logre una clasificación que sirva para que Mumuki le indique al estudiante en tiempo real sugerencias como por ejemplo: que escriba un mensaje donde se explaye más (que no sólo diga "necesito ayuda"), que el mensaje no incluya todo el código sino el fragmento de duda, etc.

7.3. Consideraciones éticas

En esta sección se consideran puntos importantes con respecto al impacto ético que pueda tener la investigación, el uso de datos y las posibles aplicaciones de nuestro trabajo. Esta sección responde a los lineamientos éticos establecidos por la Asociación Global de Lingüística Computacional[1].

- La obtención del conjunto de datos para este trabajo fue bajo consentimiento y respetando tanto los términos de uso de Mumuki como la propiedad intelectual y el derecho a la privacidad de los autores de los mensajes.
- Mumuki solicita que el acceso al conjunto de datos no sea público, con el fin de minimizar los riesgos de filtración o robo de datos, y así respetar la privacidad de los usuarios.
- El lenguaje utilizado en los mensajes del foro es el español con modismos argentinos, correspondiente a la variedad del lenguaje es-AR del BCP-47[2]. Los modelos fueron entrenados solo para esta variedad del lenguaje y su desempeño puede ser diferente para otra variedad.
- El dataset contiene mensajes de un foro de consultas para educación o distancia y los resultados pueden no generalizar a un foro de consultas que tiene clases presenciales porque los estudiantes y los docentes tienen una relación más personal. Esta diferencia de relación puede verse en las estrategias de cortesía[3] positivas y negativas usadas en los mensajes del foro, como por ejemplo *"Pedro, cómo lo ve?"* o *"Señor profesor, si no es mucha molestia, podría corregir mi programa?"*.

[1] https://2021.naacl.org/ethics/faq
[2] https://tools.ietf.org/rfc/bcp/bcp47.txt
[3] https://en.wikipedia.org/wiki/Politeness_theory

- Como se detalló en el Capítulo 4 de Metodología, los mensajes del foro del conjunto de datos fueron anonimizados por parte de Mumuki previo a la entrega del mismo, con el fin de minimizar el riesgo de divulgar información personal de los usuarios.

- En caso de que Mumuki decida de alguna manera utilizar los modelos obtenidos en este trabajo para determinar si un mensaje de un estudiante necesita atención de un mentor, el beneficio óptimo sería que el modelo clasifique un forward looking act como tal, lo que potencialmente hará que el mensaje sea atendido por un mentor. En el caso de que el modelo de clasificación haga una clasificación incorrecta, pueden ocurrir dos tipos de errores, previamente mencionados en el Capítulo 5: que un mensaje que es backward looking act sea clasificado como forward looking act, donde el posible "daño" sería desperdiciar el tiempo del docente; o que un mensaje que es forward looking act sea clasificado como backward looking act, donde el posible daño sería dejar a un mensaje de un estudiante sin respuesta de un mentor, siendo este el peor de los dos casos.

- Los modelos obtenidos en este trabajo tienen como objetivo ser potencialmente utilizados en el foro de consultas de Mumuki. El entrenamiento de estos modelos fue pensado y realizado en base a datos recolectados de Mumuki, por lo que su utilización en foros educativos en línea de otros dominios o para otro tipo de tareas no es recomendable.

- Todos los modelos de aprendizaje automatizado, incluidos los presentados en este trabajo, son vulnerables a ataques adversarios. Los ataques adversarios son datos que han sido especialmente diseñados para conseguir que se clasifiquen erróneamente. No vemos una motivación particular en esta tarea para intentar engañar al sistema. Si el estudiante necesita una respuesta a su mensaje simplemente puede hacerlo explícito en un forward looking act.

- Antes de que este sistema se ponga en producción debería tenerse especial cuidado en evaluar su desempeño sobre minorías lingüísticas. Si el sistema es usado por personas que usen otro tipo de lexemas para roles sociales. Por ejemplo, en ciertas zonas de México, en determinados contextos, "gracias" significa "no". Esto podría causar errores sistemáticos de los modelos presentados en este trabajo, que pueden clasificar con más errores mensajes de minorías lingüísticas, por ejemplo, de inmigrantes.

- Finalmente, los mensajes backward looking act cumplen un rol social que es útil para conseguir que las personas formen una relación colaborativa. La falta

de lectura de estos mensajes puede generar que la relación entre mentores y estudiantes se vea enrarecida. Por ejemplo, imaginar la situación de un mentor que nunca lee la palabra “gracias”.

Apéndice A

Conjunto de datos - Consultas

En este apéndice se presenta la estructura de la tabla de consultas del conjunto de datos provisto por Mumuki, a partir de la cual se realiza un análisis de qué campos son importantes (los cuáles están subrayados) para nuestra tarea para poder reconstruir las consultas que realizaron los estudiantes en el foro. Una consulta es una publicación que realiza un estudiante en el foro de Mumuki, en la cual idealmente da a conocer, mediante una declaración inicial, una duda o problema que está teniendo con un determinado ejercicio. Mumuki le da la posibilidad al estudiante de realizar una consulta en el foro cuando envía una solución incorrecta a un ejercicio determinado, de esta manera, cada consulta está relacionada con un ejercicio. La idea de realizar una consulta en el foro es iniciar una conversación con otros usuarios (tanto mentores como otros estudiantes) para que lo ayuden a resolver el problema. A partir de esta tabla, es posible conocer todas las consultas realizadas por los estudiantes en el foro de Mumuki y sus declaraciones iniciales, también conocidos durante el desarrollo de la tesis como *mensajes iniciales*.

- **id**: Identificador de la consulta.
- **status**: Estado de la consulta. Detallados en la Sección 3.2
- **title**: Título que el creador de la consulta le da a la misma.
- **description**: Texto del mensaje que escribió la persona cuando creó la consulta.
- **item_type**: Tipo de ejercicio que se trata.
- **item_id**: Id de ejercicio sobre el cual la consulta fue creada.
- **created_at**: Fecha de creación de la consulta.
- **updated_at**: Última vez que la consulta cambió de estado

- **upvotes_count**: Candidad de votos positivos que tiene la consulta (nivel de valoración general)
- **solution**: Código enviado como intento de resolución del ejercicio.
- **submission_status**: Una resolución de ejercicio puede tener 4 status
  - **Passed**: Es marcado por Mumuki con un color verde. Significa que la resolución es correcta, esto es: no tiene errores de sintaxis, pasa todos los casos de test y tiene en cuenta las expectativas definidas por el docente.
  - **Error**: Es marcado por Mumuki con un color rojo oscuro. La resolución tiene errores de sintaxis.
  - **Failed**: Es marcado por Mumuki con un color rojo claro. La resolución es sintácticamente correcta pero no pasa todos los casos de test
  - **Passed with warnings**: Es marcado por Mumuki con un color amarillo. La resolución es sintácticamente correcta, pasa todos los tests, pero no cumple con todas las expectativas definidas por el docente.
- **feedback**: Texto devuelto por Mumuki hacia el usuario como retroalimentación.
- **test_results**: Resultados de los tests evaluados sobre la resolución enviada.
- **submission_id**: Identificador de la resolución enviada.
- **messages_count**: Cantidad de mensajes enviados en la consulta hasta el momento.
- **validated_messages_count**: Cantidad de mensajes validados en la consulta.
- **requires_moderator_response**: Binario: Verdadero si cumple con las condiciones de filtrado de Requiere atención".
- **last_moderator_access_by_id**: La última vez que un mentor entró a la consulta.
- **last_moderator_access_at**: Fecha de la última vez que un mentor entró a la consulta.
- **status_updated_by_id**: Identificador del usuario que actualizo el estado de la consulta por última vez.
- **status_updated_at**: Fecha de la última actualización de estado que tuvo la consulta.
- **initiator_id**:Identificador de la persona que creó la consulta.

Apéndice B

Conjunto de datos - Mensajes

En este apéndice se presenta la estructura de la tabla de mensajes del conjunto de datos provisto por Mumuki, a partir de la cual se realiza un análisis de qué campos son importantes (los cuáles están subrayados) para nuestra tarea para poder reconstruir, en conjunto la tabla de consultas detallada en el Apéndice A, las conversaciones que se llevaron a cabo en el foro de Mumuki. En el Apéndice A se presenta la estructura de la tabla de consultas, a partir de la cual se puede obtener la información de todas las consultas realizadas por los estudiantes en el foro de Mumuki, y las declaraciones iniciales (también conocidas como *mensajes iniciales* durante el desarrollo de la tesis) que el estudiante creador de la consulta realiza al publicarla. A partir de la tabla de mensajes, es posible conocer todas las declaraciones realizadas en la consulta posteriores a la declaración inicial realizada por el estudiante creador de la consulta (lo que también se conoce como *mensajes no iniciales* durante el desarrollo de la tesis). De esta manera, utilizando la tabla de consultas y la tabla de mensajes, es posible obtener todos los mensajes enviados y así recrear las conversaciones del foro de consultas de Mumuki.

- **id**: Identificador del usuario que escribe el mensaje.
- **content**: Texto del mensaje
- **created_at**: Fecha de creación del mensaje
- **updated_at**: Fecha de la última vez que algún atributo del mensaje fue modificado (es válido, no es una pregunta, etc.)
- **discussion_id**: Id de la consulta a la que pertenece el mensaje.
- **approved**: Binario: Verdadero si el mensaje es validado por un mentor.
- **not_actually_a_question**: Binario: Verdadero si el mensaje no es una pregunta.

- **sender_uid**: Identificador de la persona que escribió el mensaje.

Bibliografía

[1] Andrew Adams, Tharindu Liyanagunawardena y Shirley Williams. «MOOCs: a Systematic Study of the Published Literature 2008-2012». En: *International Review of Research in Open and Distance Learning* 14 (ene. de 2013), págs. 202-227.

[2] J. Arguello y Kyle Shaffer. «Predicting Speech Acts in MOOC Forum Posts». En: *ICWSM*. 2015.

[3] J. L. Austin. «How to do things with words.» En: *Cambridge: Harvard University Press.* (1962).

[4] K. Bach y R. Harnish. «Linguistic communication and speech acts.» En: *MIT Press.* (1979).

[5] Luciana Benotti y col. «The Effect of a Web-Based Coding Tool with Automatic Feedback on Students' Performance and Perceptions». En: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. SIGCSE '18. Association for Computing Machinery, 2018, págs. 2-7. ISBN: 9781450351034.

[6] L. Breiman y col. *Classification and Regression Trees*. Taylor & Francis, 1984. ISBN: 9780412048418. URL: https://books.google.com.ar/books?id=JwQx-WOmSyQC.

[7] Leo Breiman. «Random Forests». En: *Mach. Learn.* (2001), págs. 5-32. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

[8] Peter Brusilovsky y col. «Increasing Adoption of Smart Learning Content for Computer Science Education». En: jun. de 2014. DOI: 10.1145/2713609.2713611.

[9] Nello Cristianini y Elisa Ricci. «Support Vector Machines». En: *Encyclopedia of Algorithms*. Ed. por Ming-Yang Kao. Boston, MA: Springer US, 2008, págs. 928-932. ISBN: 978-0-387-30162-4. DOI: 10.1007/978-0-387-30162-4_415. URL: https://doi.org/10.1007/978-0-387-30162-4_415.

[10] Andrew Ng Daphne Koller y Zhenghao Chen. *Retention and Intention in Massive Open Online Courses: In Depth*. URL: https://er.educause.edu/articles/2013/6/retention-and-intention-in-massive-open-online-courses-in-depth.

[11] «Gini Index». En: *The Concise Encyclopedia of Statistics*. New York, NY: Springer New York, 2008, págs. 231-233. ISBN: 978-0-387-32833-1. DOI: 10.1007/978-0-387-32833-1_169. URL: https://doi.org/10.1007/978-0-387-32833-1_169.

[12] Marcos Javier Gómez. «Aspectos de adquisición de lenguaje en la enseñanza de programación». En: (2020).

[13] Alex Graves y col. «A Novel Connectionist System for Unconstrained Handwriting Recognition». En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.5 (2009), págs. 855-868. DOI: 10.1109/TPAMI.2008.137.

[14] Mitchell Green. *Speech Acts, The Stanford Encyclopedia of Philosophy (Winter 2020 Edition), Edward N. Zalta (ed.)* URL: https://plato.stanford.edu/archives/win2020/entries/speech-acts/.

[15] H. P. Grice. «Logic and conversation. Cole, P. and Morgan, J. L. (Eds.), Speech Acts: Syntax and Semantics Volume 3"». En: *Academic Press* (1975), págs. 41-58.

[16] Ito. Hisa. *Diagrams of taxonomy of speech acts. Acta Theologica.* URL: http://www.scielo.org.za/scielo.php?script=sci_arttext&pid=S1015-87582015000200007&lng=en&tlng=en.

[17] Sepp Hochreiter y Jürgen Schmidhuber. «Long Short-term Memory». En: *Neural computation* 9 (dic. de 1997), págs. 1735-80. DOI: 10.1162/neco.1997.9.8.1735.

[18] Amal Al-Ibrahim y Hend Al-Khalifa. «Observing online discussions in educational social networks: A case study». En: *2014 International Conference on Web and Open Access to Learning, ICWOAL 2014* (ene. de 2015). DOI: 10.1109/ICWOAL.2014.7009189.

[19] India Irish y col. «PARQR: Automatic Post Suggestion in the Piazza Online Forum to Support Degree Seeking Online Masters Students». En: *Proceedings of the Seventh ACM Conference on Learning @ Scale*. L@S '20. Virtual Event, USA: Association for Computing Machinery, 2020, págs. 125-134. ISBN: 9781450379519. DOI: 10.1145/3386527.3405914. URL: https://doi.org/10.1145/3386527.3405914.

[20] Maximilian Jenders, Ralf Krestel y Felix Naumann. «Which Answer is Best?: Predicting Accepted Answers in MOOC Forums». En: abr. de 2016, págs. 679-684. DOI: 10.1145/2872518.2890567.

[21] Xin Jin y Jiawei Han. «K-Means Clustering». En: *Encyclopedia of Machine Learning*. Ed. por Claude Sammut y Geoffrey I. Webb. Boston, MA: Springer US, 2010, págs. 563-564. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_425. URL: https://doi.org/10.1007/978-0-387-30164-8_425.

[22] D. Jurafsky y J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition.* Prentice Hall Series in Artificial Intelligence. Prentice Hall, 2000. ISBN: 0130950696.

[23] Jihie Kim y Jeonhyung Kang. «Towards identifying unresolved discussions in student online forums». En: *Applied Intelligence* 40 (jun. de 2014). DOI: 10.1007/s10489-013-0481-1.

[24] S. C Kleene. «Mathematical logic». En: (1967).

[25] Pierre L., Ildiko P. y Lilja Ø David S. andMontserrat B. «Anonymisation Models for Text Data: State of the art, Challenges and Future Directions». En: *ACL* (2021).

[26] Xiangang Li y Xihong Wu. «Constructing Long Short-Term Memory based Deep Recurrent Neural Networks for Large Vocabulary Speech Recognition». En: *CoRR* abs/1410.4281 (2014). arXiv: 1410.4281. URL: http://arxiv.org/abs/1410.4281.

[27] W. Loh. «Fifty Years of Classification and Regression Trees 1». En: 2014.

[28] Pablo E Martínez López y col. «The GOBSTONES method for teaching computer programming». En: *2017 XLIII Latin American Computer Conference (CLEI)*. 2017, págs. 1-9.

[29] Amit Mandelbaum y Adi Shalev. *Word Embeddings and Their Use In Sentence Classification Tasks*. 2016. arXiv: 1610.08229 [cs.LG].

[30] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997. ISBN: 978-0-07-042807-2.

[31] Mehryar Mohri, Rostamizadeh Afshin y Ameet Talwalkar. *Foundations of Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, 2018. ISBN: 978-0-262-03940-6.

[32] Marco Moresi. «Modelado automático de trayectorias de aprendizaje: ¿Cuándo generar ayuda personalizada para principiantes en programación?» En: (2018).

[33] John Neter, William Wasserman y Michael H Kutner. «Applied linear regression models.» En: (1989).

[34] Blikstein P. y col. «Programming pluralism: Using learning analytics to detect patterns in the learning of computer programming.» En: *Journal of the Learning Sciences, 23(4)* (2014), págs. 561-599.

[35] Razvan Pascanu, Tomas Mikolov y Yoshua Bengio. *On the difficulty of training Recurrent Neural Networks*. 2013. arXiv: 1211.5063 [cs.LG].

[36] Fabian Pedregosa y col. «Scikit-Learn: Machine Learning in Python». En: *The J. Mach. Learn. Res.* 12 (nov. de 2011), págs. 2825-2830. ISSN: 1532-4435.

[37] Massimo Poesio y David Traum. «Towards an Axiomatization of Dialogue Acts». En: (mayo de 1998).

[38] Sebastian Raschka. Packt Publishing, Limited, 2019, págs. 37-38. ISBN: 9781789958294.

[39] Tobias Rohloff, Dominic Sauer y Christoph Meinel. «Students' Achievement of Personalized Learning Objectives in MOOCs». En: *Proceedings of the Seventh ACM Conference on Learning @ Scale*. L@S '20. Virtual Event, USA: Association for Computing Machinery, 2020, págs. 147-156. ISBN: 9781450379519. DOI: 10.1145/3386527.3405918. URL: https://doi.org/10.1145/3386527.3405918.

[40] V. Rus y col. «Automatic Discovery of Speech Act Categories in Educational Games». En: *EDM*. 2012.

[41] Stuart Russell y Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010.

[42] Schegloff Sacks y Jefferson. «A simplest systematics for the organization of turn-taking for conversation.» En: (1974), págs. 696-735.

[43] S. Salzberg. «C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993». En: *Machine Learning* 16 (1994), págs. 235-240.

[44] «Expectation-Maximization Algorithm». En: *Encyclopedia of Machine Learning*. Ed. por Claude Sammut y Geoffrey I. Webb. Boston, MA: Springer US, 2010, págs. 387-387. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_291. URL: https://doi.org/10.1007/978-0-387-30164-8_291.

[45] E. A. Schegloff. «A simplest systematics for the organization of turn-taking for conversation.» En: *American Anthropologist 70* (1968), págs. 1075-1095.

[46] J. Searle. «Speech Acts: An Essay in the Philosophy of Language.» En: *Cambridge: Cambridge University Press.* (1969).

[47] A. M. Turing. «Computing machinery and intelligence». En: *Mind* LIX.236 (oct. de 1950), págs. 433-460. ISSN: 0026-4423. DOI: 10.1093/mind/LIX.236.433. URL: https://doi.org/10.1093/mind/LIX.236.433.

[48] A. M. Turing. «On Computable Numbers, with an Application to the Entscheidungsproblem». En: *Proceedings of the London Mathematical Society* s2-42.1 (ene. de 1937), págs. 230-265. ISSN: 0024-6115. DOI: 10.1112/plms/s2-42.1.230. eprint: https://academic.oup.com/plms/article-pdf/s2-42/1/230/4317544/s2-42-1-230.pdf. URL: https://doi.org/10.1112/plms/s2-42.1.230.

[49] L. Wang y col. «Applied linear regression models.» En: *In Proceedings of the Fourth (2017) ACM Conference on Learning@Scale* (2017), págs. 201-204.

[50] Geoffrey I. Webb. «Naïve Bayes». En: *Encyclopedia of Machine Learning*. Ed. por Claude Sammut y Geoffrey I. Webb. Boston, MA: Springer US, 2010, págs. 713-714. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_576. URL: https://doi.org/10.1007/978-0-387-30164-8_576.

[51] L. Wittgenstein. «Philosophical Investigations.» En: *Blackwell.* (1953).

[52] Michael M. Wolf. «Mathematical Foundations of Supervised Learning». En: (2020).

[53] Meng Xia y col. «Using Information Visualization to Promote Students' Reflection on "Gaming the System" in Online Learning». En: *Proceedings of the Seventh ACM Conference on Learning @ Scale*. L@S '20. Virtual Event, USA: Association for Computing Machinery, 2020, págs. 37-49. ISBN: 9781450379519. DOI: 10.1145/3386527.3405924. URL: https://doi.org/10.1145/3386527.3405924.