Universidad Nacional del Centro de la Provincia de Buenos Aires

Facultad de Ciencias Exactas

Ingeniería de Sistemas

Tesis de Grado

Framework de Modelado de Usuarios utilizando técnicas de Deep Learning a partir de Big Data

Autor: Francisco Serrano

Directores

Ing. Franco D. Berdun

Dr. Marcelo G. Armentano

Índice de Contenidos

Capítulo 1: Introducción 12
1.1. Motivación 13
1.2. Objetivos 14
1.3. Organización del Informe 14
Capítulo 2: Estado del Arte 16
2.1. Modelado de Usuarios 16
2.1.1. Interaction Process Analysis 17
2.1.2. Symlog 18
2.1.3. Trabajos Relacionados 22
2.2. Deep Learning 23
2.2.1. Redes Neuronales Planas 25
2.2.2. Redes Neuronales Convolucionales (CNN) 30
2.2.3. Redes Neuronales Recurrentes (RNN) 32
2.2.4. Trabajos Relacionados 35
2.3. Conclusión 39
Capítulo 3: Desarrollo de la Propuesta 41
3.1. Estructura de la Herramienta 41
3.1.1. Front-End 42
3.1.2. Back-End 44
3.2. Proceso Predictivo 46
3.2.1. Pipeline 47
3.2.2. Estructura del Clasificador 48
3.3. Enfoques Propuestos 49
3.3.1. Representación de la Información 49
3.3.2. Técnicas a Emplear 51
3.4. Conclusión 52
Capítulo 4: Implementación de la Herramienta 54
4.1. Requerimientos Funcionales 54
4.2. Arquitectura de la Solución 55
4.2.1. Clasificación 56
4.2.2. Persistencia 58

4.2.3. Gestión de Formatos y Generación de Perfiles 58
4.2.4. Broker 61
4.2.5. Aplicación Web 65
4.3. Descripción de Endpoints 67
4.3.1. Clasificación 67
4.3.2. Persistencia 68
4.3.3. Gestión de Formatos y Generación de Perfiles 69
4.3.4. Broker 70
4.3.5. Cliente 71
4.4. Tecnologías Empleadas 72
4.4.1. Spring Boot 72
4.4.2. Flask 74
4.4.3. Pandas 75
4.4.4. Keras 76
4.4.5. SQLite 77
4.4.6. MongoDB 78
4.4.7. Angular 78
4.4.8. Docker 79
4.5. Aplicación de Tecnologías 79
4.5.1. Clasificación 80
4.5.2. Persistencia 82
4.5.3. Gestión de Formatos y Generación de Perfiles 82
4.5.4. Broker 83
4.5.5. Aplicación Web 84
4.6. Escenarios de Calidad 85
4.6.1. Flexibilidad 85
4.6.2. Escalabilidad 86
4.6.3. Usabilidad 87
4.6.4. Portabilidad 88
4.6.5. Interoperabilidad 88
Capítulo 5: Enfoques de Deep Learning 90
5.1. Redes Neuronales Planas 91
5.2. Redes Neuronales Convolucionales (CNN) 94

5.3. Redes Neuronales Recurrentes (RNN) 97
5.4. Redes Neuronales Convolucionales Recurrentes (CRNN) 100
5.5. Conclusiones 104
Capítulo 6: Experimentación 106
6.1. Configuración 106
6.2. Resultados con Enfoques de Aprendizaje 110
6.2.1. Tablas correspondientes al enfoque basado en Redes Densas 111
6.2.2. Tablas correspondientes al enfoque basado en Redes Convolucionales 115
6.2.3. Tablas correspondientes al enfoque basado en Redes Recurrentes 117
6.2.4. Tablas correspondientes al enfoque basado en Redes Convolucionales Recurrentes . 120
6.3. Análisis de Métricas de Usuarios 122
6.4. Conclusión 140
Capítulo 7: Conclusiones 142
7.1. Implicancias 143
7.2. Limitaciones 144
7.3. Trabajos Futuros 144
7.3.1. Enriquecimiento de Interacciones 144
7.3.2. Edición en línea de Conjuntos de Mensajes 145
7.3.3. Detección Automática de Roles de Equipo 145
Anexo A 146
A.1. Enfoque 1: Redes Planas 146
A.2. Enfoque 2: Redes Neuronales Convolucionales (CNN) 149
A.3. Enfoque 3: Redes Neuronales Recurrentes (RNN) 152
A.4. Enfoque 4: Redes Neuronales Convolucionales Recurrentes (CRNN) 155
Bibliografía 158

Índice de Figuras

2.1. Mapeo de cada pregunta con sus respectivos indicadores Symlog 20

2.2. Ejemplo 1 de obtención de valores resultantes para Symlog 21

2.3. Ejemplo 2 de obtención de valores resultantes para Symlog 21

2.4. Diagrama de disposición de dimensiones Symlog 22

2.5. Función lineal que se ajusta a los ejemplos provistos 24

2.6. Función logística o sigmoide 26

2.7. Proceso de elaboración de predicciones en una regresión logística 27

2.8. Abstracción de una regresión logística 27

2.9. Arquitectura de una red neuronal con dos capas 28

2.10. Gráfico de la función Rectifier Linear Unit (ReLU) 29

2.11. Ejemplo de convolución para tensores de dos dimensiones 30

2.12. Ejemplo de operación de convolución para detección de bordes 31

2.13. Arquitectura de la red neuronal convolucional LeNet 31

2.14. Ejemplo de Max-Pooling, que toma una ventana de 2x2, y un stride de 2 32

2.15. Ejemplo de celda recurrente 33

2.16. Celda LSTM 34

2.17. Celda GRU 35

2.18. Ejemplo de word-embeddings 36

2.19. Ejemplo de una red con una capa recurrente bidireccional 37

2.20. Ejemplo de overfitting en donde el modelo está sobre ajustado al dataset 39

3.1. Pantalla de la aplicación con el aspecto de la interfaz provista 42

3.2. Visualización del grupo de indicadores correspondiente a las reacciones 43

3.3. Reacciones de la tabla para un determinado integrante 44

3.4. Tipo de formato de mensaje con el cual trabajará el framework 45

3.5. Ilustración abstracta del proceso predictivo que se lleva a cabo 47

3.6. Obtención de los indicadores Symlog a partir de las conductas IPA 48

3.7. El mensaje es sometido a un clasificador de reacciones 49

3.8. Ejemplo ilustrativo de representación de documentos 50

3.9. Arquitectura típica de las Redes Neuronales 52
4.1. Diagrama de arquitectura de la aplicación web 56
4.2. Proceso predictivo de conductas 57
4.3. Ejemplo de generación de perfiles para una conversación en formato ARFF 60
4.4. Flujo de ejecución básico empleando un CSV como fuente de clasificación 62
4.5. Flujo de ejecución básico empleando un ARFF como fuente de clasificación 63
4.6. Ejemplo de la carga de información a través de los distintos tipos de fuentes 66
4.7. Ejemplo ilustrativo de mapeo automático de las clases por parte de Spring 73
4.8. Ejemplo de endpoint desarrollado en Python, empleando Flask 75
4.9. Ejemplo de la facilidad de integrar CSV con Pandas y SQL 76
4.10. Ejemplo de red neuronal convolucional empleando Keras 77
4.11. Apariencia de la aplicación web 85
5.1. Metodología utilizada para el desarrollo de los modelos predictivos 90
5.2. Arquitectura del primer enfoque empleando redes neuronales densas 92
5.3. Métricas resultantes del entrenamiento del enfoque más básico 94
5.4. Arquitectura del segundo enfoque empleando CNN 95
5.5. Métricas resultantes del entrenamiento del enfoque de CNN 97
5.6. Arquitectura del tercer enfoque empleando RNN 98
5.7. Métricas resultantes del entrenamiento del enfoque de RNN 100
5.8. Arquitectura del cuarto enfoque empleando CRNN 101
5.9. Métricas resultantes del entrenamiento del enfoque de CRNN 103
5.10. Comparación de las métricas de los cuatro enfoques 104
6.1. Tarjeta en donde se provee el dataset con formato ARFF 123
6.2. Gráfico con las conductas predichas por la red neuronal en su mejor enfoque 124
6.3. Gráfico con las reacciones resultantes de agrupar las conductas 125
6.4. Gráfico con los indicadores Symlog 126
6.5. Gráfico con las dimensiones Symlog 127
6.6. Gráfico de conductas a lo largo de las sesiones que tuvo el usuario 128
6.7. Gráfico con las reacciones producto de agrupar las conductas expuestas 129

6.8. Gráfico con los seis indicadores Symlog 130
6.9. Dimensiones Symlog obtenidas 131
6.10. Gráfico con las conductas predichas por la red neuronal (CRNN) 132
6.11. Gráfico de reacciones generado a partir de las conductas 133
6.12. Gráfico con los seis indicadores Symlog obtenidos a partir del mapeo 134
6.13. Dimensiones Symlog obtenidas a partir de la agrupación de indicadores 135
6.14. Tabla 1 con los indicadores de conducta 136
6.15. Tabla 1 con los indicadores de reacciones 136
6.16. Tabla 1 con los indicadores de dimensión Symlog 137
6.17. Tabla 2 con los indicadores de conducta 137
6.18. Tabla 2 con los indicadores de reacción 138
6.19. Tabla 2 con los indicadores de dimensión Symlog 138
6.20. Tabla 3 con los indicadores de conducta 139
6.21. Tabla 3 con los indicadores de reacciones 139
6.22. Tabla 3 con los indicadores de dimensión Symlog 140

Índice de Tablas

2.1. Composición de las dimensiones Symlog a partir de los indicadores 18

2.2. Conductas establecidas por el Interaction Process Analysis 19

4.1. Exposición HTTP del microservicio encargado de efectuar la clasificación 67

4.2. Exposición HTTP del microservicio encargado de la persistencia 68

4.3. Exposición HTTP del servicio encargado de los formatos y armado de perfiles 69

4.4. Exposición HTTP del servicio encargado de interactuar con los demás servicios 71

5.1. Hiperparámetros utilizados para el entrenamiento del enfoque más básico 93

5.2. Hiperparámetros utilizados para el entrenamiento del enfoque CNN 96

5.3. Hiperparámetros utilizados para el entrenamiento del enfoque RNN 99

5.4. Hiperparámetros utilizados para el entrenamiento del enfoque CRNN 102

5.5. Asociación entre la identificación de la Figura 5.10 con cada enfoque 105

5.6. Análisis del mejor enfoque para cada función 105

6.1. Distribución de conductas habiendo aplicado balanceo de clases 108

6.2. Distribución de reacciones habiendo aplicado balanceo de clases 109

6.3. Matriz de confusión para la dimensión U/D (up/down) 111

6.4. Matriz de confusión para la dimensión P/N (positive/negative) 112

6.5. Matriz de confusión para la dimensión F/B (forward/backward) 112

6.6. Métricas obtenidas a partir de las matrices de confusión asociadas a U/D 113

6.7. Métricas obtenidas a partir de las matrices de confusión asociadas a P/N 114

6.8. Métricas obtenidas a partir de las matrices de confusión asociadas a F/B 114

6.9. Valores de accuracy para cada dimensión 114

6.10. Matriz de confusión para la dimensión U/D (up/down) 115

6.11. Métricas obtenidas a partir de las matrices de confusión asociadas a U/D 115

6.12. Matriz de confusión para la dimensión P/N (positive/negative) 115

6.13. Métricas obtenidas a partir de las matrices de confusión asociadas a P/N 116

6.14. Matriz de confusión para la dimensión F/B (forward/backward) 116

6.15. Métricas obtenidas a partir de las matrices de confusión asociadas a F/B 116

6.16. Valores de accuracy para cada dimensión 117

6.17. Matriz de confusión para la dimensión U/D (up/down) 117
6.18. Métricas obtenidas a partir de las matrices de confusión asociadas a U/D 117
6.19. Matriz de confusión para la dimensión P/N (positive/negative) 118
6.20. Métricas obtenidas a partir de las matrices de confusión asociadas a P/N 118
6.21. Matriz de confusión para la dimensión F/B (forward/backward) 118
6.22. Métricas obtenidas a partir de las matrices de confusión asociadas a F/B 119
6.23. Valores de accuracy para cada dimensión 119
6.24. Matriz de confusión para la dimensión U/D (up/down) 120
6.25. Métricas obtenidas a partir de las matrices de confusión asociadas a U/D 120
6.26. Matriz de confusión para la dimensión P/N (positive/negative) 120
6.27. Métricas obtenidas a partir de las matrices de confusión asociadas a P/N 121
6.28. Matriz de confusión para la dimensión F/B (forward/backward) 121
6.29. Métricas obtenidas a partir de las matrices de confusión asociadas a F/B 121
6.30. Valores de accuracy para cada dimensión 121
6.31. Tabla de comparación de los enfoques desarrollados 141
A.1. Matriz asociada a la presencia de dominancia 146
A.2. Matriz asociada a la presencia del intermedio entre dominancia y sumisión 146
A.3. Matriz asociada a la presencia de sumisión 146
A.4. Matriz asociada a la presencia "amistoso" 147
A.5. Matriz asociada a la presencia del intermedio entre "amistoso" y "no amistoso" 147
A.6. Matriz asociada a la presencia de "no amistoso" 147
A.7. Matriz asociada a la presencia de "orientación a la tarea" 148
A.8. Matriz asociada a la presencia del intermedio entre "orientación a la tarea" y "orientación a lo socioemocional" 148
A.9. Matriz asociada a la presencia del indicador de "orientación a lo socioemocional" 148
A.10. Matriz asociada a la presencia del indicador de dominancia 149
A.11. Matriz asociada a la presencia del intermedio entre dominancia y sumisión 149
A.12. Matriz asociada a la presencia del indicador de sumisión 149
A.13. Matriz asociada a la presencia del indicador de "amistoso" 150

A.14. Matriz asociada a la presencia del intermedio entre "amistoso" y "no amistoso" 150

A.15. Matriz asociada a la presencia del indicador de "no amistoso" 150

A.16. Matriz asociada a la presencia del indicador de "orientación a la tarea" 151

A.17. Matriz asociada a la presencia del intermedio entre "orientación a la tarea" y "orientación a lo socioemocional" 151

A.18. Matriz asociada a la presencia del indicador de "orientación a lo socioemocional" 151

A.19. Matriz asociada a la presencia del indicador de dominancia 152

A.20. Matriz asociada a la presencia del intermedio entre dominancia y sumisión 152

A.21. Matriz asociada a la presencia del indicador de sumisión 152

A.22. Matriz asociada a la presencia del indicador de "amistoso" 153

A.23. Matriz asociada a la presencia del intermedio entre "amistoso" y "no amistoso" 153

A.24. Matriz asociada a la presencia del indicador de "no amistoso" 153

A.25. Matriz asociada a la presencia del indicador de "orientación a la tarea" 154

A.26. Matriz asociada a la presencia del intermedio entre "orientación a la tarea" y "orientación a lo socioemocional" 154

A.27. Matriz asociada a la presencia del indicador de "orientación a lo socioemocional" 154

A.28. Matriz asociada a la presencia del indicador de dominancia 155

A.29. Matriz asociada a la presencia del intermedio entre dominancia y sumisión 155

A.30. Matriz asociada a la presencia del indicador de sumisión 155

A.31. Matriz asociada a la presencia del indicador de "amistoso" 156

A.32. Matriz asociada a la presencia del intermedio entre "amistoso" y "no amistoso" 156

A.33. Matriz asociada a la presencia del indicador de "no amistoso" 156

A.34. Matriz asociada a la presencia del indicador de "orientación a la tarea" 157

A.35. Matriz asociada a la presencia del intermedio entre "orientación a la tarea" y "orientación a lo socioemocional" 157

A.36. Matriz asociada a la presencia del indicador de "orientación a lo socioemocional" 157

Capítulo 1: Introducción

El análisis de la dinámica de grupos es extremadamente útil para entender y predecir el desempeño de equipos de trabajos, puesto que, en este contexto, pueden surgir naturalmente problemas de colaboración. La inteligencia artificial, y especialmente las técnicas de aprendizaje de máquina, permiten automatizar el proceso de observación y análisis de grupos de usuarios que utilizan una plataforma colaborativa en línea.

La dinámica de grupos se define como el proceso de interacción en un grupo para resolver una determinada tarea [1]. Un enfoque para estudiar la dinámica de grupos se basa en el análisis de las interacciones entre los miembros de un equipo para extraer información útil sobre el comportamiento de los mismos. Uno de los modelos más utilizados en la literatura es el método IPA [2], que consiste en asociar una categoría a cada interacción. Cada categoría indica un tipo de conducta, de forma tal que un mensaje que envíe un usuario, como por ejemplo “me parece bien”, se asocia con una categoría que indica “muestra acuerdo”. Luego a partir de IPA, Bales propone SYMLOG [3] para incluir el comportamiento no verbal en el modelo. Dicho comportamiento es caracterizado a través de una serie de valores que caracterizan a las tres dimensiones propuestas por el modelo.

A partir de los métodos que trabajan sobre la dinámica de grupos, es posible y necesario capturar el comportamiento que cada individuo posee en un grupo de colaboración, es decir, en un grupo cuyos integrantes deben trabajar colaborativamente para resolver una tarea. Dicho comportamiento consiste en las interacciones que poseen los miembros, las cuales pueden manifestarse en forma de *mensajes*. Esto tiene especial relevancia al querer analizar la dinámica de grupos de trabajo en contextos distribuidos, en donde el empleo de plataformas de comunicación juega un rol primordial en el desarrollo de la tarea asignada a un determinado grupo.

Muchas de las plataformas de comunicación online disponibles hoy en día (por ejemplo, Skype[1], Hangouts[2], Facebook[3] o Slack[4]), proporcionan la información necesaria para capturar el comportamiento de los usuarios a partir de los mensajes que cada usuario envía. Contar con una herramienta que facilite dicha captura abriría puertas para proveer una asistencia inteligente en el armado de grupos de trabajo, así como también se podría facilitar la detección de comportamientos conflictivos, que afecten la colaboración durante la resolución de los diversos problemas que se presenten.

Normalmente, tanto las tareas de armado de grupos de trabajo como la de detección de comportamientos conflictivos, se realizan de forma manual [4, 5], lo cual constituye una tarea sumamente tediosa. En ambos casos los investigadores recurrieron a una *categorización manual* de

[1] https://www.skype.com/es/

[2] https://hangouts.google.com/

[3] http://www.facebook.com/

[4] https://slack.com/intl/es/

las interacciones observadas de forma directa, siendo en estos casos grabaciones de video. Si bien dicho procedimiento es viable para una pequeña cantidad de interacciones, en casos de interacciones mediante mensajes escritos, dicha tarea se complica de sobremanera debido a la cantidad de interacciones con las cuales se trabaja. De esta manera, en caso de poder categorizar dichas interacciones de forma automática, se podría propiciar la observación de las interacciones para grandes cantidades de datos.

En los últimos años, el campo del aprendizaje automático (subrama de inteligencia artificial) ha progresado consistentemente debido a dos factores: el crecimiento de los volúmenes de información disponible, y el crecimiento del poder de cómputo que se ha dado en los últimos años. Este contexto estimuló la proliferación de las técnicas de aprendizaje profundo, que se caracterizan por ser más potentes que las técnicas tradicionales. Si bien estas técnicas se basan en una teoría que tiene más de treinta años, el empleo de la misma era inviable en aquél entonces debido a un problema de poder de cómputo. Hoy en día el contexto es otro, por lo que la problemática de capturar el comportamiento de los usuarios constituye una oportunidad sumamente interesante para aplicar este tipo de técnicas.

1.1. Motivación

Los perfiles colaborativos indican la forma en la que un individuo se comporta con el resto de los miembros de un grupo de colaboración. Por lo tanto, una correcta identificación de los mismos podría propiciar diversas cuestiones como, por ejemplo, la elaboración de asistencias automáticas para el armado de grupos. Además, en caso de que un perfil colaborativo pudiese capturar correctamente los aspectos negativos en el comportamiento de un individuo, se podrían realizar también inferencias automáticas de situaciones conflictivas. De esta manera, se evita recaer puramente en la observación manual de las interacciones por parte de un supervisor. Esto se traduciría a que, por ejemplo, en caso de combinar la labor de un supervisor con una asistencia automatizada, la resolución del armado de grupos podría mejorarse de forma considerable.

Esto último es de especial relevancia en el contexto de las empresas de software, donde a partir de las características de cada integrante, se pueden generar grupos de trabajos que maximicen la eficiencia al momento de resolver una determinada tarea. Por ejemplo, en grupos de trabajo distribuidos, donde existen equipos de empleados ubicados en diferentes regiones, cualquier conocimiento previo del perfil de cada uno de los empleados podría permitir una mejor organización de los grupos de desarrollo.

Normalmente este tipo de tareas se realiza a través de encuestas o de observación directa sobre la forma en que los miembros interactúan. Por ejemplo, en Jira[5], una plataforma online de gestión de proyectos, se suelen proveer encuestas a llenar por los miembros de un equipo, para así conocer las características colaborativas de los mismos. Generalmente las encuestas no constituyen un procedimiento del todo alegre, ya que, para realizar correctamente la inferencia, dichas encuestas suelen ser consideradamente extensas. La posibilidad de automatizar esta inferencia mediante

[5] https://es.atlassian.com/software/jira

técnicas de aprendizaje de máquina, evitaría el empleo de encuestas para la captura de comportamientos, por lo que los perfiles colaborativos se podrían generar de forma automática, es decir sin esfuerzo de los usuarios o miembros del equipo.

1.2. Objetivos

La propuesta que se realiza en este trabajo de grado consiste en la elaboración de un framework que facilite la generación de perfiles colaborativos de usuario, siguiendo el modelo SYMLOG, a partir de las interacciones de los mismos en un determinado espacio compartido. Dicha generación de perfiles se realiza a través del uso de técnicas de aprendizaje profundo, más precisamente a partir del empleo de diferentes conceptos inherentes a las redes neuronales. Se experimentará con varios conceptos de redes neuronales, para luego formar un modelo que integre dichos conceptos, a fin de maximizar la precisión con la que se generan los perfiles en cuestión.

Con el objetivo de validar los perfiles generados, se evaluarán cuatro enfoques de aprendizaje profundo, los cuales varían en cuanto al concepto que emplean. Mientras el primero consiste en básicamente emplear capas densas para la arquitectura de la red neuronal, el segundo emplea capas basadas en la operación de convolución, y el tercero emplea operaciones recurrentes. Por último, el cuarto enfoque toma lo mejor de estos conceptos a fin de generar una arquitectura de red neuronal híbrida, que mejore las métricas obtenidas en enfoques anteriores.

Además del framework, se implementará una aplicación web a fin de organizar gráficamente la información que se obtiene. De esta forma, se acerca el uso de este tipo de herramientas a usuarios con mínimos conocimientos en sistemas de este tipo. Como el framework está pensado para trabajar como backend, la información que el mismo proporciona, se muestra en el cliente a través de diversos gráficos y tablas. De esta manera se facilita el entendimiento de la misma, evitando tener que recaer en la lectura de formatos como por ejemplo JSON o CSV.

El proceso de elaboración de perfiles colaborativos comenzará con un conjunto de mensajes provenientes del diálogo generado por los miembros de un grupo trabajando colaborativamente para resolver una tarea común. Dichos mensajes serán sometidos a los modelos de redes neuronales para así obtener una categorización de cada uno de ellos, según el método IPA. Luego a partir de dichas categorías, se establecerán correspondencias entre el método IPA y SYMLOG. Es decir que se efectúa un mapeo de las categorizaciones obtenidas por las redes, al trinomio correspondiente a cada dimensión del modelo SYMLOG.

1.3. Organización del Informe

En esta sección se proveen detalles acerca de cómo se estructura el presente informe, ofreciendo una descripción introductoria de las temáticas abordadas en cada uno de los siete capítulos que componen al presente trabajo de grado.

El presente capítulo, es decir el Capítulo 1, se corresponde con la introducción al trabajo de grado. Se trata de un capítulo corto que provee una noción básica de los conceptos e ideas que se emplearán a lo largo de todo el informe.

El Capítulo 2 está abocado a explicar detalladamente y discutir los aspectos teóricos que se emplean en el trabajo de grado. No se proveen detalles relacionados al trabajo concreto que se realiza, sino que se explica exclusivamente la teoría independientemente de la aplicación de la misma. El énfasis está puesto en el modelado de usuarios y en el aprendizaje profundo, realizando un recorrido en profundidad de cada uno de los aspectos inherentes a cada eje del trabajo de grado.

En el Capítulo 3 se presentan aspectos relacionados puramente con el trabajo de grado, en donde se expone en profundidad el desarrollo de la propuesta. Se aborda el desarrollo del trabajo, en donde se abarcan cuestiones tanto de la implementación de la herramienta, como del modelo predictivo, que constituye el foco del trabajo. El modelo predictivo se comenta de punto a punto, es decir desde la representación de la información, hasta la generación de los perfiles de usuario. La propuesta no cubre detalles concretos de la implementación, sino que se plantea desde un plano abstracto. Si bien se muestra la forma en que se va a resolver el problema presentado, no se hace mención alguna a las tecnologías a emplear.

Luego en el Capítulo 4, se explica en detalle uno de los temas centrales del trabajo de grado: la implementación. A partir del desarrollo de la propuesta, en este capítulo se hace especial énfasis en la forma en que se va a llevar a cabo la implementación. Esto es, la aplicación web y su correspondiente arquitectura subyacente. Además, se hace mención a las tecnologías empleadas, junto con las características distintivas de cada una, y la forma en que son aprovechadas en la implementación. Con respecto a los modelos predictivos, se hace referencia a los servicios que contienen la funcionalidad en cuestión, aunque no se mencionan los aspectos inherentes a la implementación de cada enfoque.

Los enfoques de redes neuronales, son detallados con profundidad en el Capítulo 5. Dicha profundidad se relaciona con en el entrenamiento de las mismas, efectuando un barrido por la arquitectura, los hiperparámetros, y las métricas que arroja cada enfoque durante el entrenamiento. Los enfoques desarrollados emplean cada uno un concepto distinto por detrás, por lo que se comenzará con el más básico, para finalizar con el más complejo de los cuatro.

Con respecto a la experimentación, la misma es abordada, en el Capítulo 6, a partir de la implementación realizada. En este proceso, se comienza explicando las características del conjunto de datos empleado, para luego realizar dos tipos de ensayos. En el primero relacionado al framework, en donde se toman diversas métricas relacionada a cuan bien se generan los perfiles de usuario. Luego en el segundo se efectúa una experimentación que consiste en la obtención de diversos perfiles de usuario, pero a través de la aplicación web desarrollada.

Por último, en el Capítulo 7, se efectúa una conclusión general del trabajo de grado, a fin de resaltar los aspectos más importantes, así como también mencionar las limitaciones encontradas. Además, se especifican aquellas cuestiones cuyo alcance estuvo por fuera de este trabajo, que sirven de puntapié para el desarrollo de futuros trabajos, o bien para el desarrollo de nuevas líneas de investigación.

Capítulo 2: Estado del Arte

En este capítulo se presenta y explica el contenido teórico que se emplea en este trabajo de grado. Se abarcan tres ejes, en donde cada uno es abordado en su respectiva sección. El mayor énfasis está en la teoría de redes neuronales con respecto a las técnicas de aprendizaje profundo; y en SYMLOG, por parte de la teoría de modelado de usuarios.

En la sección 2.1, se explica la teoría referida al modelado de usuarios. Se emplearon dos teorías, siendo ambas desarrolladas por R. F. Bales. Si bien se explican ambas en profundidad, el énfasis se ubica sobre Symlog, que provee el marco teórico que más uso se le va a dar en el trabajo de grado. Además, se realiza un relevamiento de trabajos relacionados con la temática, tanto a nivel teórico como a nivel práctico.

En la Sección 2.2. se realiza un barrido general por los aspectos teóricos más importantes del aprendizaje profundo. Se comienza analizando la estructura de la red neuronal más básica, para luego progresar en la explicación del tema, pasando por arquitecturas más complejas como las redes neuronales convolucionales, y las redes neuronales recurrentes. También se hace mención a enfoques híbridos entre distintos tipos de arquitecturas, poniendo a disposición los beneficios que conlleva. Al igual que en la sección anterior, una vez introducida toda la base teórica de aprendizaje profundo, se efectúa un relevamiento de aquellos trabajos relacionados más importantes del área, que motivaron la implementación llevada a cabo en este trabajo, la cual se explica en profundidad en el Capítulo 5.

Luego en la tercer sección, se realiza una breve introducción sobre los conceptos más importantes de Big Data. En esta sección se analizan aquellos aspectos que están puramente relacionados con el trabajo de grado, pero no se aborda el tema en profundidad, ya que el eje está puesto en el modelado de usuarios y en las técnicas de aprendizaje profundo.

2.1. Modelado de Usuarios

Empresas, universidades, y otras organizaciones están tomando cada vez mayor conciencia de la necesidad de personalizar los servicios ofrecidos para usuarios o grupos de usuarios. Para poder ofrecer información personalizada, es necesario monitorear el comportamiento de un usuario, y realizar generalizaciones y predicciones basadas en esas observaciones [6]. La organización de la información sobre el usuario que puede ser obtenida de esta manera se denomina modelado *de usuario* [7], que constituye una subárea de HCI (Human-Computer Interaction), que investiga el diseño y uso de la tecnología aplicada en la interacción de un humano con una computadora, como su nombre bien indica.

Modelar un usuario puede consistir en simplemente capturar características básicas como por ejemplo el género, o bien puede llevar a cabo a tareas complejas como por ejemplo descubrir conocimiento experto, de forma similar a la que un especialista podría clasificar un conjunto de datos. Los sistemas de modelado de usuarios pueden adquirir información de forma explícita o implícita. De forma explícita puede ser a través de por ejemplo encuestas, cuestionarios, etc. De forma implícita puede realizarse a través de observación del comportamiento del usuario en

determinados sitios, monitoreo de los lugares en donde interactúa (mapas de calor), o por ejemplo la forma en que juega un juego. Si bien las posibilidades son muy extensas, el objetivo generalmente es efectuar inferencias y/o predicciones del comportamiento de un usuario, a fin de realizar recomendaciones, abstraer procedimientos rutinarios, etc. Para el ejemplo de la predicción de la forma en la que un usuario juega a un determinado videojuego, se podría aprovechar la captura de información implícita para poder desarrollar algoritmos de inteligencia artificial más eficientes para que puedan jugar contra el usuario, a fin de proveer una experiencia de juego mucho más real.

El presente trabajo de grado parte del concepto de la dinámica de grupos [8], en el contexto de plataformas digitales colaborativas. Dicho concepto hace referencia a los componentes y procesos existentes en cualquier grupo. En un sentido estricto, puede referirse a una herramienta que apunte a regular o monitorear las interacciones dentro de un grupo [9]. Para un contexto colaborativo dado, los integrantes de un determinado grupo deben interactuar de forma tal que el resultado final de una tarea a realizar, sea lo mejor posible.

Sin embargo, cada individuo naturalmente tiene su propio comportamiento, el cual, al manifestarse dentro de un grupo, puede influir positiva o negativamente en el desarrollo de la tarea en cuestión. Para ello, el modelado de usuarios puede consistir en estructurar el conocimiento que se tiene de un usuario. De esta manera, teniendo una estructura que representa una observación del usuario, es posible realizar diversos análisis sobre dicha estructura a fin extraer información nueva sobre el dominio. En el caso del presente trabajo de grado, se emplean diversas técnicas del área del modelado de usuarios para generar perfiles de los mismos a partir de sus interacciones dentro de un grupo de trabajo. Dichos perfiles proveen la posibilidad de que, por ejemplo, ante cualquier tipo de conflicto dentro de la comunicación en un grupo de trabajo, el mismo pueda ser detectado. Otro tipo de posibilidad que surge es la formación de grupos. Conociendo las características de cada individuo a partir de técnicas de modelado de usuarios, es posible elaborar una disposición de grupos más eficiente, ubicando a cada integrante con aquellos con los cuales mejores resultados obtendrían.

2.1.1. Interaction Process Analysis

En 1950 R.F. Bales publicó Interaction Process Analysis [2] (comúnmente denominado IPA), que consiste en un framework teórico en donde se asigna una determinada categoría a cada interacción de un individuo. Este tipo de categorías indica una determinada conducta por parte del remitente, en donde son doce las conductas posibles (Tabla 2.1). Dentro de estas doce conductas, se puede observar que algunos tipos de conductas están relacionadas. Un ejemplo de eso son las conductas C4 y C5, que indican “Da sugerencia” y “Da información” respectivamente. Esto da lugar a que ciertas conductas se agrupen en distintas reacciones, siendo la reacción “Responde” la que agrupa a las conductas ejemplificadas. En la Tabla 2.1 se muestra un listado de las doce conductas establecidas por Bales, junto con las reacciones que agrupan a dichas conductas.

Conductas	Reacciones
C1. Muestra solidaridad	Positiva
C2. Muestra relajamiento	
C3. Muestra acuerdo	
C4. Da sugerencia	Responde
C5. Da opiniones	
C6. Da información	
C7. Pide información	Pregunta
C8. Pide opinión	
C9. Pide sugerencia	
C10. Muestra desacuerdo	Negativa
C11. Muestra tensión	
C12. Muestra antagonismo	

Tabla 2.1. Conductas establecidas por el Interaction Process Analysis, junto con las reacciones que las agrupan.

Como se puede observar, cada interacción que un individuo posee, según IPA se categoriza con una conducta que va de C1 a C12. A su vez, cada conducta se mapea con una determinada reacción. En caso de quererse detectar situaciones de conductas conflictivas, es importante que se monitoreen los casos en donde los mensajes que un individuo envía, se categorizan con conductas pertenecientes a la reacción "Negativa", mientras que en caso de querer obtener mayor detalle del tipo de problema que surja, se deben monitorear la cantidad de interacciones categorizadas como C10, C11 o C12.

2.1.2. Symlog

La teoría Symlog [3], también ideada por R. F. Bales distingue tres dimensiones estructurales en interacciones grupales: estado, atracción y orientación de objetivos. Estas tres dimensiones en su conjunto definen *perfiles de colaboración*. La primera dimensión analiza la actitud dominante

(codificada como U) o sumisa (D) de quienes interactúan; la segunda estudia la tendencia positiva (P) o negativa (N); y finalmente, la tercera dimensión analiza la cuestión de si las personas están involucradas con la tarea (F) o con comportamientos socio-emocionales (B). A fin de facilitar el entendimiento sobre los indicadores y las dimensiones, se expone a continuación en la Tabla 2.2 la asociación entre los mismos. Cada dimensión contempla además una posible conducta neutral, logrando de esta forma 27 combinaciones. A pesar de que Symlog fue desarrollado como una extensión del modelo IPA, estas dos teorías se complementan.

Indicadores Symlog	Dimensiones Symlog
Dominante (U)	Up/Down (U/D)
Sumiso (D)	
Positivo (P)	Positive/Negative (P/N)
Negativo (N)	
Orientación a la Tarea (F)	Forward/Backward (F/B)
Orientación a lo Socioemocional (B)	

Tabla 2.2. Composición de las dimensiones Symlog a partir de los indicadores.

Para obtener un perfil SYMLOG, el proceso comienza con una encuesta que debe responder un usuario, la cual contiene una serie de 26 preguntas. Cada pregunta tiene tres respuestas posibles, “Nunca”, “A veces” y “Siempre”, en donde cada respuesta tiene una ponderación asociada, -1, 0 y 1, respectivamente. Para determinar los valores de cada dimensión, se tienen los seis indicadores descritos anteriormente: “dominante” y “sumiso”, que componen la dimensión U/D; “positivo” y “negativo”, que componen la dimensión P/N; y “orientación a tarea” y “orientación a lo socioemocional”, que componen la dimensión F/B. Cada pregunta, está asociada con uno o más de estos seis indicadores, por ejemplo, la pregunta 7 está asociada con los indicadores UNB, que se refieren a los indicadores “dominante”, “negativo”, y “orientación a lo socioemocional”. Luego, por ejemplo, si una persona responde “siempre” en esa pregunta, se suma un punto a esos tres indicadores. En la Figura 2.1 se expone una tabla con la asociación de cada pregunta con sus respectivos indicadores.

Nro. Pregunta	Código SYMLOG	Característica
1	U	Activo, dominante, habla mucho
2	UP	Extrovertido, positivo
3	UPF	Líder de tareas democrático, con propósito
4	UF	Manager orientado a negocios, asertivo
5	UNF	Autoritario, controlador, desaprobador
6	UN	Dominante, mente dura, poderoso
7	UNB	Provocativo, egocéntrico, se muestra mucho
8	UB	Hace chistes, expresivo, dramático
9	UPB	Entretenido, sociable, sonriente, cálido
10	P	Amigable, igualitario
11	PF	Trabaja de forma cooperativa con el resto
12	F	Analítico, orientado a tareas, resuelve problemas
13	NF	Legalista, hace las cosas bien
14	N	Poco amigable, negativista
15	NB	Irritable, cínico, no coopera
16	B	Muestra sentimientos y emociones
17	PB	Afectivo, querible, con quien da gusto estar
18	DP	Mira a los demás, agradecido, confiable
19	DPF	Gentil, acepta tomar responsabilidad
20	DF	Obediente, trabaja sumisamente
21	DNF	Se auto castiga, trabaja demasiado duro
22	DN	Depresivo, triste, resentido, repulsivo
23	DNB	Alienado, renuncia, abandona
24	DB	Miedo a intentar, duda de su habilidad
25	DPB	Tranquilamente contento al estar con el resto
26	D	Pasivo, introvertido, habla poco

Figura 2.1. Mapeo de cada pregunta con sus respectivos indicadores Symlog.

Una vez que se responden todas las preguntas de la encuesta, el resultado es una cantidad de puntos para cada uno de los seis indicadores. Por ejemplo, si el indicador de “dominante” tiene 6 puntos, y el de “sumiso” posee 2, entonces la dimensión U/D tendrá un valor resultante de 4 puntos. Cabe aclarar, que, al ser cada indicador de una dimensión, el opuesto al otro, los puntos correspondientes a los indicadores de conductas negativas, se restan a los puntos correspondientes a conductas positivas. Un segundo ejemplo sería si se tiene 0 puntos en el indicador de “orientación a la tarea” y -2 en el indicador de “orientación a lo socioemocional”, la dimensión F/B tiene un valor de 2 puntos. Para las tres dimensiones se procede de forma análoga.

Por último, se tiene un último aspecto para el cálculo de los valores Symlog. La teoría considera también que un exceso en el valor de cada dimensión constituye una situación de alerta, sobre todo en casos donde se da un exceso en la dimensión U/D. En dicha situación se puede inferir que la persona en cuestión es, o muy dominante, o muy sumisa. Para poder determinar este tipo de cuestiones de forma unívoca, se define un llamado “intervalo de neutralidad”, establecido en el intervalo cerrado [-3, 3]. Si los valores finales de cada dimensión caen dentro de ese intervalo, indica una situación neutral, es decir que indica ausencia de extremos. Por ejemplo, si para la dimensión U/D un individuo obtuvo un valor de 4, indica un extremo hacia la dominancia, mientras, un valor de -4 indica un extremo hacia la sumisión.

De esta manera, en caso de presentarse un extremo, se muestra la letra correspondiente al indicador que se recae fuera del intervalo. En caso de tenerse un 4 para U/D, indica una D que infiere un extremo dominante. Por ejemplo, un valor final de “UB”, indica un extremo hacia la dominancia, hacia lo socioemocional, pero para la dimensión P/N no se presentaron extremos. A modo de clarificar este proceso, en la Figura 2.2 y 2.3 se muestran ejemplos de cálculo de estos valores.

Indicadores Symlog	Valor	Dimensiones	Valor	Recae por fuera de [3, -3] ?	Valor Resultante	Valor Resultante Agrupado
Dominante (U)	6	Up/Down (U/D)	3	SI	U	UB
Sumiso (D)	3					
Positivo (P)	2	Positive/Negative (P/N)	1	NO		
Negativo (N)	1					
Orientación a la Tarea (F)	2	Forward/Backward (F/B)	-4	SI	B	
Orientación a lo Socioemocional (B)	6					

Figura 2.2. Ejemplo 1 de obtención de valores resultantes para Symlog.

Indicadores Symlog	Valor	Dimensiones	Valor	Recae por fuera de [3, -3] ?	Valor Resultante	Valor Resultante Agrupado
Dominante (U)	0	Up/Down (U/D)	1	NO		F
Sumiso (D)	-1					
Positivo (P)	2	Positive/Negative (P/N)	1	NO		
Negativo (N)	1					
Orientación a la Tarea (F)	2	Forward/Backward (F/B)	4	SI	F	
Orientación a lo Socioemocional (B)	-2					

Figura 2.3. Ejemplo 2 de obtención de valores resultantes para Symlog.

Como se puede observar en éstos últimos dos ejemplos, cada uno de ellos se corresponde con un individuo distinto. Estos valores surgen producto de las respuestas que ingresaron en las respectivas encuestas. El valor final que caracteriza a cada individuo, es el de la última columna, siendo “UB” para el primer ejemplo, y “F” para el segundo ejemplo.

Habiendo explicado la nomenclatura para caracterizar las dimensiones Symlog, ahora se puede comprender correctamente el origen de las 27 combinaciones posibles mencionadas anteriormente. En la Figura 2.4 se muestra un diagrama tridimensional con estas combinaciones.

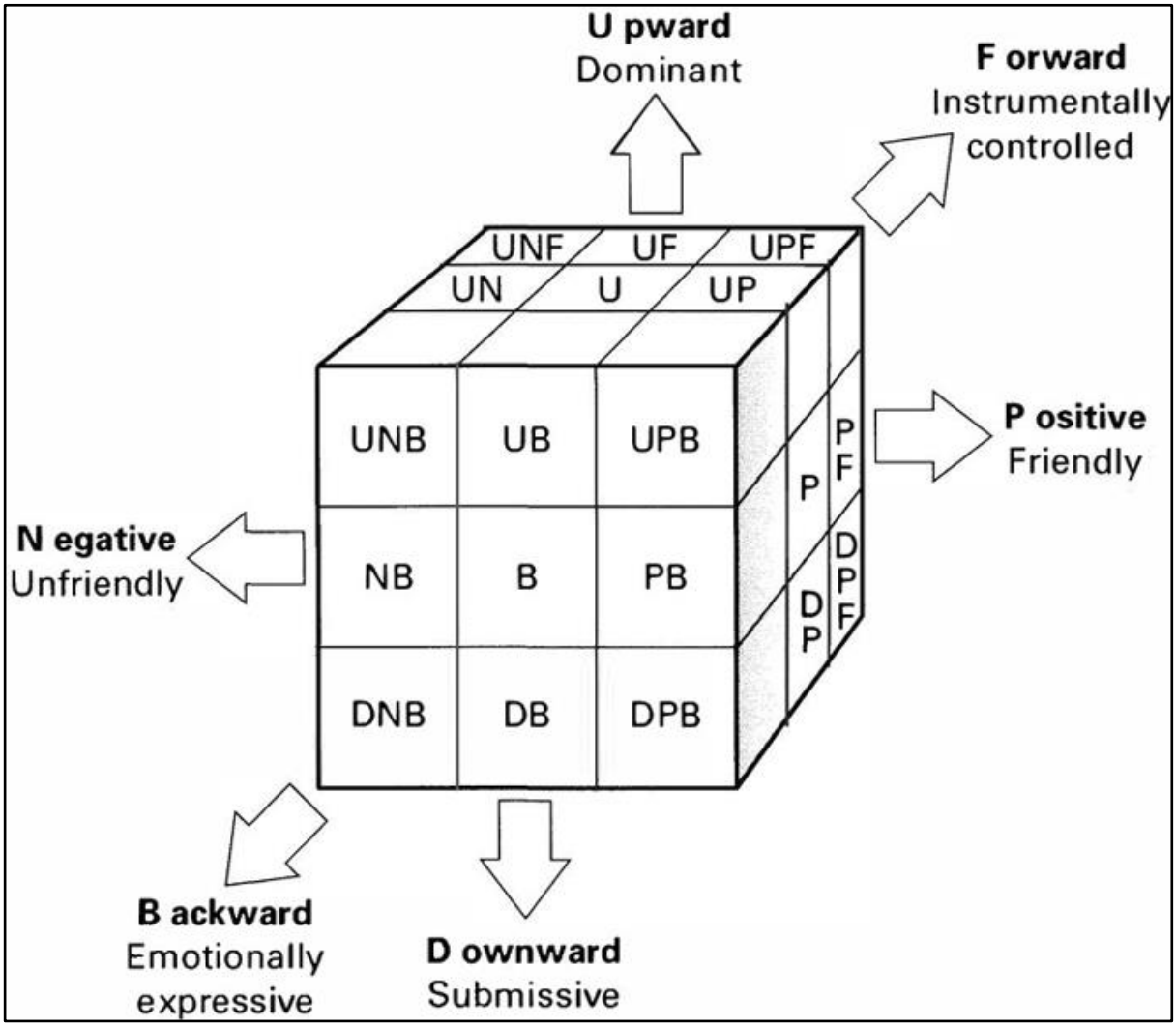

Figura 2.4. Diagrama tridimensional que grafica la disposición de las tres dimensiones de Symlog.

2.1.3. Trabajos Relacionados

A continuación, se detallan algunos de los trabajos encontrados en la literatura que emplean las teorías recientemente expuestas. Estos trabajos se relacionan y motivan al presente trabajo de grado a fin de proveer nuevas ideas para la resolución de los problemas que surjan, o para extender/perfeccionar el conocimiento sobre el dominio.

A partir de la teoría IPA de Bales para el análisis de interacciones, se experimentó con el estudio de reuniones laborales, estudiando en detalle las distintas formas de interacción involucradas en las mismas y generando reportes individuales sobre el comportamiento de los participantes, incluyendo información cualitativa sobre el funcionamiento de la dinámica grupal [11]. Con esta información, se propone aumentar el grado de conocimiento de los grupos sobre su propia dinámica con el fin de mejorar la efectividad del trabajo colaborativo. Este mismo objetivo se buscó también, de forma exitosa, en investigaciones realizadas por Berdún et al. [12], en donde a partir de

interacciones manifestadas a través de texto libre, se empleó la teoría IPA para efectuar predicciones asociadas a cada mensaje. Dichos mensajes provienen de un contexto de CSCW (Computer-Supported Collaborative Work), en donde un grupo de personas puede trabajar en conjunto de forma digital, para alcanzar la realización de una tarea dada.

Siguiendo en la línea de trabajo colaborativo, también hubo trabajos relacionados a modelado automático de perfiles, siendo en este caso la teoría Symlog la salida de la predicción [13], mientras que en el anterior se emplearon las categorizaciones IPA como salida. En este caso, a través de un juego online colaborativo, se tienen varias características que se asocian a cada usuario. Luego a partir de las acciones observadas, estados y contextos, se desarrollaron modelos predictivos que generan las dimensiones Symlog asociadas a cada usuario.

Otro trabajo enmarcado en esta tendencia es el Mood Meter System [14]. En él, se parte del interés por capturar los aspectos socio-emocionales de la interacción grupal, desarrollando diversos gráficos para representar las fluctuaciones en el estado de ánimo de los participantes a través del tiempo. El trabajo se basa en las dimensiones establecidas en el modelo Symlog de Bales para llevar a cabo esta caracterización. Las calificaciones de "estado de ánimo" de los participantes se agrupan en una única calificación del grupo, que se representa de forma esquemática mediante círculos concéntricos o estrellas de color y densidad variable; se representan grupos divergentes de imágenes dispersas y grupos convergentes por las imágenes concentradas. Una desventaja presente en esta herramienta es la gran dependencia en la calidad de los esquemas desarrollados para mapear la participación y los estados de ánimos al modelo Symlog; los datos requieren un proceso de interpretación para ser traducidos al modelo desarrollado por Bales, de la cual depende la precisión de la descripción a la que se arriba.

Symgroup [15], introduce la idea de agentes sociales, es decir, agentes cuya meta es mantener o mejorar el contexto social de un grupo, en esta temática. Se trata de una herramienta para la discusión aumentada a través del empleo de tres agentes sociales; uno de ellos, un agente que implementa Symlog, que analiza la conducta de los participantes de los experimentos, mientras que los dos restantes implementan teorías ad-hoc sobre las actitudes de cada participante sobre la discusión. En esta línea de introducción de agentes en la evaluación de comportamiento grupal, Ellis, Wainer y Barthelmess [16] presentan el Proyecto Neem, cuya finalidad es introducir agentes inteligentes y participantes virtuales en un entorno de reuniones distribuidas. Dichos agentes incorporan conocimiento sobre distintos aspectos de lo que hace a una reunión "buena", incorporando modelos de cómo una reunión debería proceder, e interviniendo en ella cuando es preciso corregir su curso, todo ello apoyándose en los fundamentos teóricos del modelo Symlog.

2.2. Deep Learning

En esta sección se realiza un barrido completo sobre los aspectos más relevantes del aprendizaje profundo, una de las subramas del aprendizaje automático, que a su vez es una subrama de inteligencia artificial. Se divide la teoría en un orden de complejidad progresivo, comenzando por los temas más básicos, como lo son las redes neuronales planas, para luego abordar temas más complejos e interesantes como las redes convolucionales y recurrentes. Por último, también se

comentan los métodos híbridos, que toman lo mejor de la convolución y la recurrencia, para así cubrir los aspectos necesarios para citar diversos trabajos relacionados en el área.

En un problema de aprendizaje automático típico, básicamente lo que se realiza es una función en la que, a partir de una determinada entrada X, devuelve una salida Y. Esta salida Y consiste en una probabilidad de que X pertenezca a una determinada categoría. Por ejemplo, una función lineal, puede utilizarse como una predicción. Si bien un predictor basado en una regresión lineal no suele ser la mejor opción a emplear, éste resulta ser el modelo base con el que se inicia el aprendizaje en Deep Learning.

En la Figura 2.5 se muestra un ejemplo de regresión lineal. Como se podrá observar, un problema de aprendizaje automático tradicional supervisado, necesita una serie de datos etiquetados, sobre los cuales, en este caso, se plantea la función lineal que mejor se adapte a esos datos. Esta función lineal se obtiene mediante la búsqueda de la mejor combinación de la pendiente y la ordenada al origen. La pendiente y la ordenada se llaman *parámetros*, ya que constituyen los valores que deben modificarse para que la función lineal se pueda ajustar de la mejor manera a los datos provistos.

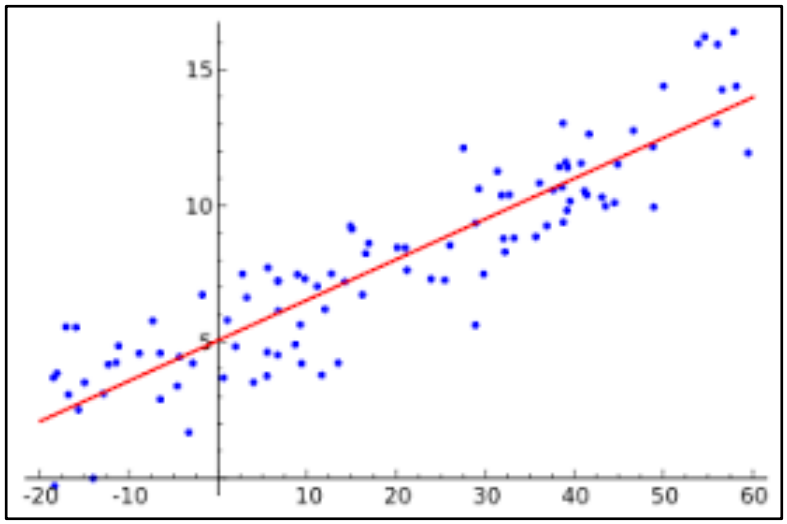

Figura 2.5. Ejemplo de función lineal (rojo) que se ajusta a las instancias provistas (azul).

La mejor función lineal que se pueda obtener para estos datos, es aquella que minimice la distancia de los puntos a la misma recta, es decir, aquella que minimice el error. Esto se debe a que, en caso de querer efectuar una predicción, la misma se va a llevar a cabo sometiendo la entrada a la función lineal resultante.

El proceso que se lleva a cabo para obtener la mejor función lineal es básicamente el siguiente. Inicialmente se tiene un par <pendiente, ordenada> aleatorio, el cual obviamente estará muy lejos de la función lineal óptima. Los puntos azules son sometidos a la función, y para cada punto, se compara la predicción arrojada por la función, contra el valor real de dicho punto. Es decir, el valor en Y del punto azul (el valor real), contra el valor en Y que retorna la función. La diferencia que exista entre el valor real, y el predicho, constituye el error en un punto. La función que calcula el error para

un solo ejemplo del dataset, es decir para un solo punto azul, es la *función de pérdida*. A partir de las funciones de pérdida para todos los puntos, se puede armar la *función de costo*, que resulta en el promedio de las funciones de pérdida para todo el dataset.

Esta función de costo, es la que define el "aprendizaje" de nuestro problema. La función de costo es una función convexa, en donde a partir de dos valores, retorna un valor. Es decir que, proporcionándole el valor de la pendiente y de la ordenada, retorna el error que posee la función lineal representada por esos dos parámetros. La clave de un problema de aprendizaje automático es encontrar un par <pendiente, ordenada> tal que el error que produzcan sea mínimo, es decir que hay que minimizar la función de costo, a través de la búsqueda de un mínimo global en dicha función, en caso de ser posible.

Esto se realiza a través de una técnica llamada gradiente descendente. El gradiente es un vector que se calcula a través de las derivadas parciales en un determinado punto de la función, y que retorna un vector que indica la dirección de mayor crecimiento. En el método del gradiente descendente, lógicamente, se invierte al vector gradiente obtenido en un punto (dado por la ordenada y la pendiente) para obtener la dirección de mayor decrecimiento. De esta manera, se pueden actualizar los valores de la pendiente y de la ordenada según este gradiente:

$$valor_{nuevo} = valor_{viejo} - \alpha * gradiente$$

De esta manera, los parámetros se actualizan según esta fórmula. La letra alfa que se puede observar, es un valor que regula cuánto desciende el valor por la función de costo, y es conocido como la tasa de aprendizaje (en inglés *learning rate*). Con la fórmula provista, los valores de la pendiente y la ordenada se ven modificados la cantidad de veces necesaria hasta que el error obtenido sea mínimo: esta sencilla idea es el *aprendizaje* de un modelo. Como se podrá observar en la Figura 2.5, la función que se muestra toma un escalar como entrada. Este mismo concepto de gradiente descendente se puede tomar para entradas de n-dimensiones, y para cualquier otro tipo de función, sea logística, cuadrática, polinómica, etc. Lo que variará en caso de usar otras funciones o cantidad de dimensiones, es la cantidad de parámetros que se deberán utilizar para optimizar para minimizar la función de costo. De esta manera se puede concluir que, un problema de aprendizaje profundo, se trata básicamente de minimizar una función de costo. Es decir que dado un conjunto de puntos de n-dimensiones, se debe encontrar una función que se ajuste de la mejor manera a dichos puntos. De la mejor manera no quiere decir que se ajuste perfectamente: debe tener un error mínimo ya que, como el problema es la predicción de puntos, ante la provisión de un punto que no formaba parte del dataset original, el mismo debe poderse predecir correctamente.

2.2.1. Redes Neuronales Planas

Hasta el momento se vio el funcionamiento de la regresión lineal. Existe otro método de aprendizaje que se llama *regresión logística*, que a nivel matemático consiste en aplicar una determinada

función llamada función sigmoide o logística, a la salida de la función lineal. En la Figura 2.6 se muestra un ejemplo de la apariencia de la función logística.

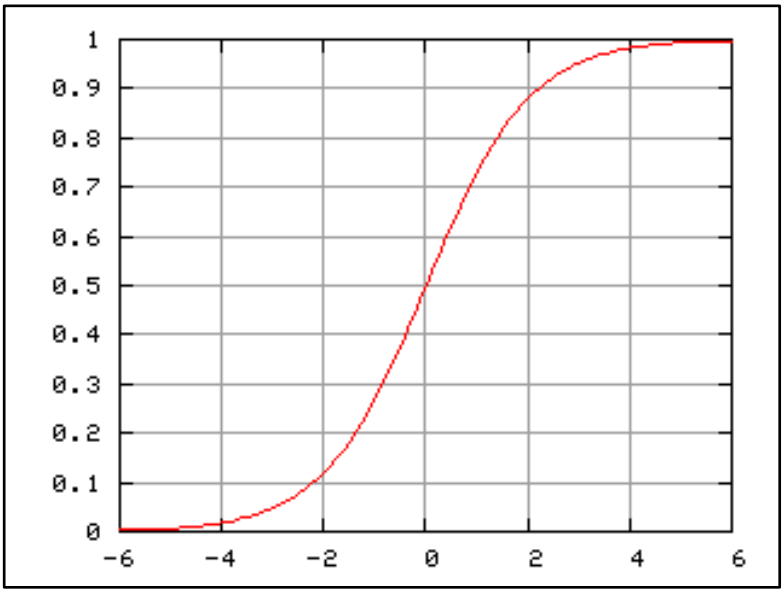

Figura 2.6. Función logística o sigmoide, que se aplica en la regresión logística sobre el resultado de la función lineal.

Como se puede observar, una función de ese estilo es capaz de efectuar predicciones con mayor precisión que una regresión lineal, lo cual constituye ser un mejor predictor, aunque un poco más complejo de entender. En la Figura 2.7 se muestra la parte en donde la regresión logística se diferencia de la lineal. Un detalle a aclarar es que el proceso de elaboración de la predicción en una red neuronal se suele llamar “propagación hacia adelante” (en inglés Forward-Propagation), ya que el flujo de los datos va “hacia delante”. Cuando se calculan las derivadas para actualizar los parámetros, se llama “propagación hacia atrás”, más conocido como Backpropagation. Este proceso fue ideado por Geoffrey Hinton et. al [17] en 1986, que consistió en un trabajo de investigación fundacional para toda la teoría de aprendizaje profundo. Si bien estos conceptos de propagación no se suelen nombrar mucho en la regresión logística, es importante nombrarlos a esta altura para ir familiarizándose con el término, a fin de facilitar el entendimiento del mismo al momento de ver redes neuronales concretas.

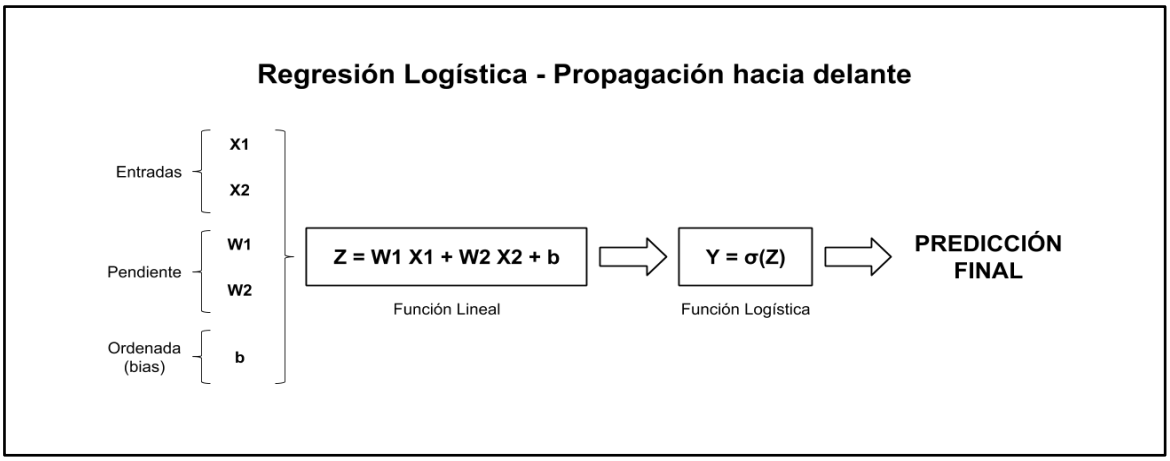

Figura 2.7. Proceso de elaboración de predicciones en una regresión logística.

Este proceso que se puede observar, de la regresión logística, constituye el proceso de cómputo que realiza lo que se llama como una "neurona", la unidad básica de una red neuronal. Es decir que el cálculo que se ve en la Figura 2.7, se abstrae en una neurona. Una red neuronal es entonces, naturalmente, un conjunto de neuronas. A fin de estandarizar la representación de las redes neuronales, una regresión logística se puede abstraer de la siguiente manera, como se muestra en la Figura 2.8.

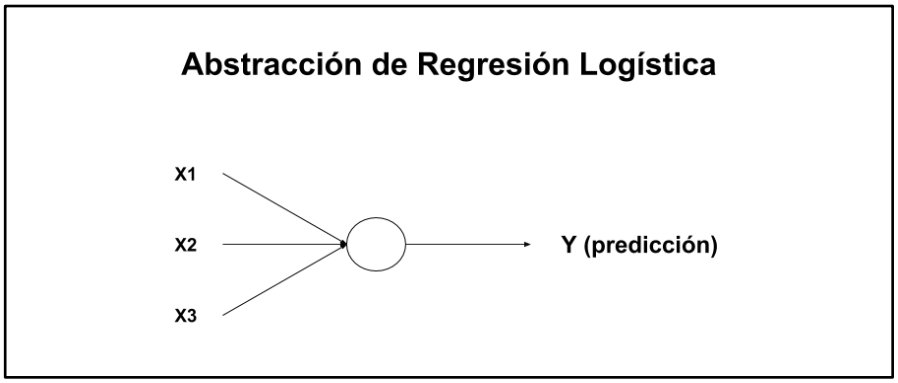

Figura 2.8. Abstracción de una regresión logística.

De esta manera, como una red neuronal se puede definir como un conjunto de neuronas, con una determinada disposición, se provee en la Figura 2.9 un ejemplo de una arquitectura básica de redes neuronales. Como se puede observar, se realizaron dos grandes modificaciones con respecto al ejemplo anterior. Se tienen dos capas, en donde la primer capa, tiene tres neuronas, mientras que la última tiene una sola. La segunda gran diferencia radica en que, si bien la salida de la neurona consistía en la predicción final, en este caso, para la primer captura, la salida de las mismas alimenta la última neurona, de la última capa.

A modo de recapitulación, cada neurona, tiene una determinada cantidad de parámetros, una porción de ellos relacionada a la multiplicación que se efectúa sobre la entrada (en la función lineal sería la pendiente), y la otra relacionada a lo que se llama el bias, que consiste en la suma que se efectúa sobre el resultado del producto reciente. En cada unidad, es decir en cada neurona, se aplica una *función de activación* al resultado de esta última suma, como por ejemplo la recientemente vista función logística o sigmoide. En caso de existencia de superposición de capas, la entrada que tiene la neurona, no es la instancia provista sino la salida de la capa anterior. Es decir, que la entrada de la segunda capa es el resultado de la función de activación efectuada en la capa anterior. Y así sucesivamente para n cantidad de capas, hasta obtener la predicción final, producto de la salida de la última neurona en la última capa de la red neuronal. Este proceso de obtención de la predicción, como se nombró anteriormente, se conoce como Forward Propagation. Mientras que el proceso de actualización de los parámetros de las neuronas, se inicia a partir del cálculo del error en función de la predicción obtenida. Y a partir de la predicción, se actualizan los parámetros hacia atrás, es decir, comenzando desde la última capa, hasta la primera. Esto se debe al cálculo del gradiente, en donde se aplica la regla de la cadena. De esta manera, se obtiene una propagación hacia atrás, más conocido como *Backpropagation*.

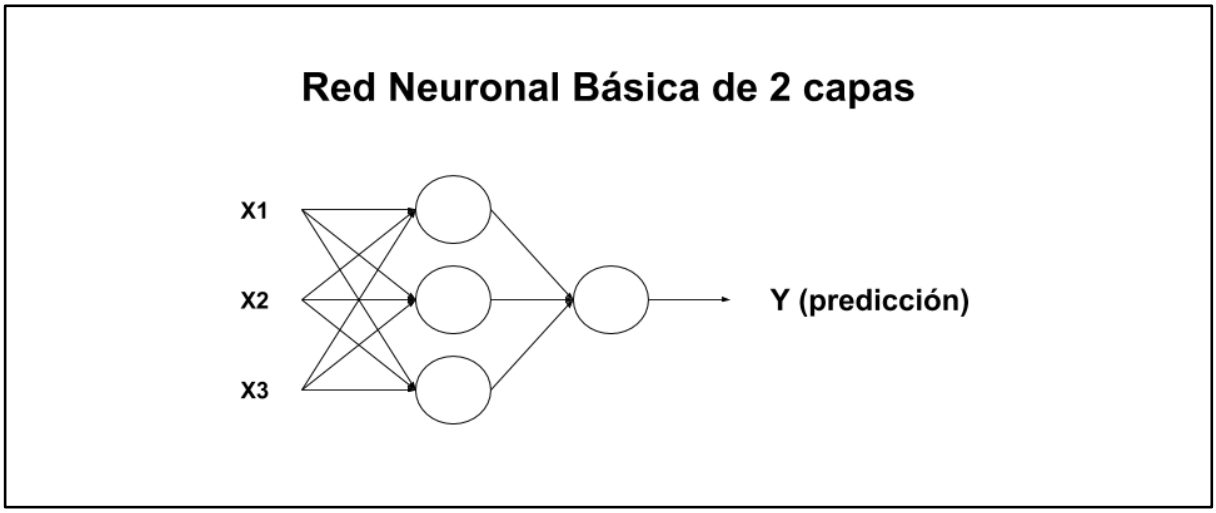

Figura 2.9. Arquitectura de una red neuronal con dos capas.

Otro aspecto a destacar de este tipo de redes, es la elección de las funciones de activación. Hasta el momento se nombró a la función logística o sigmoide, que se aplica sobre la multiplicación y suma que se efectúa en cada neurona. Sin embargo, éstas no son las únicas funciones que se suelen utilizar para el desarrollo de las redes neuronales. Existe otra muy utilizada que se llama *función de rectificación*, más comúnmente conocida como ReLU, cuya función se muestra a continuación en la Figura 2.10.

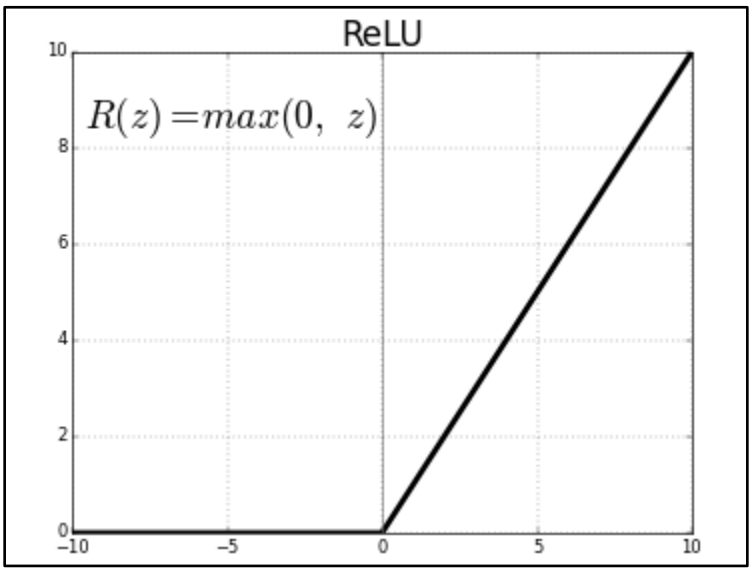

Figura 2.10. Gráfico de la función Rectifier Linear Unit (ReLU).

Esta función resulta ser la función por defecto a utilizar para las *hidden layers*, es decir para las que no son la capa de salida. La razón de ser de esta función radica en una cuestión de cómputo. Backpropagation es una teoría que fue publicada hace ya muchos años, que en su momento era difícil de implementar debido a que el cálculo de las derivadas empleando la regla de la cadena, era muy costoso computacionalmente. En estos últimos años, este proceso cobró importancia debido a que la potencia de cómputo de los equipos actuales generó que sea viable llevar a cabo este proceso. Sin embargo, las redes neuronales siguen teniendo cierta dificultad para ser entrenadas, sobre todo cuando crece demasiado la arquitectura de las mismas (debido a que el dominio lo requiere). Es por esto, y por la necesidad de acelerar el aprendizaje de las redes neuronales, que se emplea este tipo de función, que se caracteriza en asignar un valor cero para números negativos. Es decir, que cuando la entrada es cero, su derivada también es cero, por lo tanto, se torna innecesario el cálculo de la derivada, y se ahorra mucho tiempo de cómputo, lo cual redunda en un entrenamiento de la red más efectivo.

Si bien se suele utilizar ReLU para las hidden layers, para la última capa, es decir para la clasificación final, se utiliza una función sigmoide, ya que, por las características de la función, distribuye mejor la probabilidad. En caso de clasificación binaria, si la salida de la última capa es mayor a 0.5, se predice una clase, mientras que si es menor se predice la otra. Sin embargo, en caso de tener que predecirse múltiples categorías, la función sigmoide puede no ser la mejor alternativa. En caso de querer efectuar clasificaciones de varias categorías, el intervalo sobre el cual está definido la función sigmoide (0, 1), habría que separarlo uniformemente en tantos subintervalos como categorías se quieran predecir, lo cual no arroja los mejores resultados.

Luego, para casos de clasificaciones de más de dos categorías, se utiliza como función en la última capa, la *función Softmax*, que consiste en una generalización de la función logística, pero para n cantidad de categorías. Mientras que originalmente la función logística retorna un escalar con la probabilidad, esta función retorna un vector con tantos elementos como categorías se quieren

predecir. La particularidad de este vector es que cada elemento en la posición n, constituye la probabilidad de que la instancia pertenezca a la clase n. De esta manera, la clase final predicha, se toma del elemento con mayor probabilidad.

2.2.2. Redes Neuronales Convolucionales (CNN)

La convolución es una operación que constituye el bloque básico de una red neuronal convolucional. En dicha operación, se tiene un tensor de entrada (arreglo multidimensional), y se lo somete a otro tensor, llamado filtro, para producir un tensor de salida. Normalmente se emplea este tipo de operación sobre imágenes, en donde por ejemplo al querer invertir los colores de la misma, se aplica un filtro sobre la imagen. En la Figura 2.11 se provee un ejemplo de la apariencia que tiene una operación de convolución para tensores de dos dimensiones.

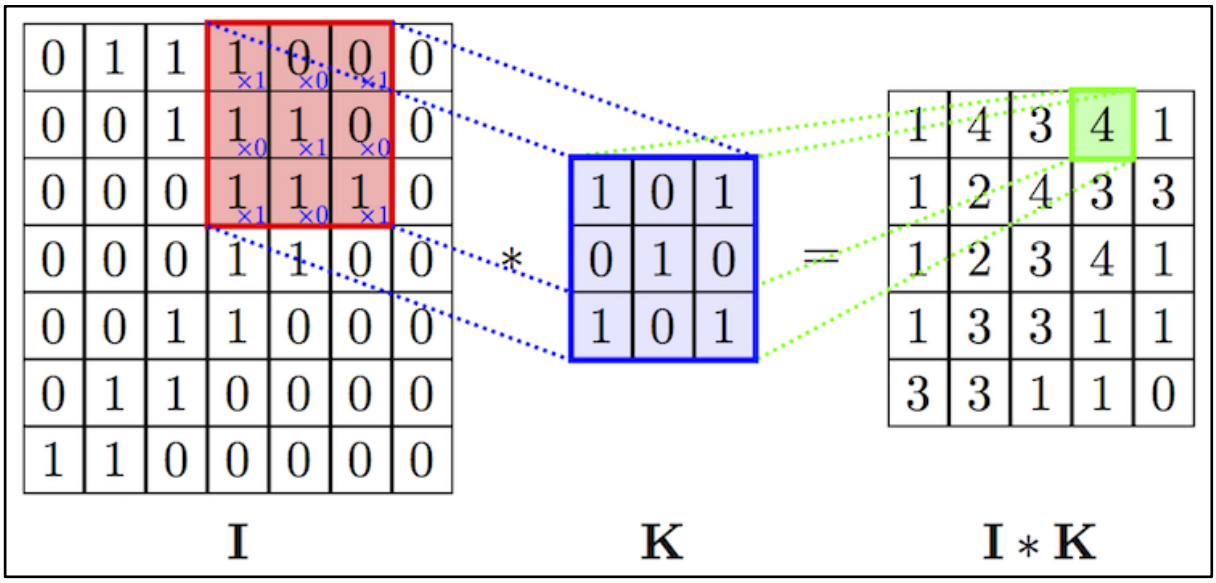

Figura 11. Ejemplo de convolución para tensores de dos dimensiones.

Como se puede observar en el ejemplo, se tiene una "ventana", que se marca en rojo, que constituye en este caso una submatriz, en donde se multiplica elemento a elemento con el filtro (azul), y a los resultados de cada multiplicación se los suma para producir el número que está en verde. Inicialmente la ventana inicia ubicada en el extremo superior izquierdo del tensor de entrada, y una vez que se efectúa la convolución, se corre la ventana una posición a la derecha, para luego realizar el mismo cálculo, pero para producir el siguiente valor del tensor de salida. Esta operación se caracteriza por producir un determinado efecto sobre la imagen, es decir que pueden detectar ciertos patrones. En la Figura 2.12 se muestra otro ejemplo de convolución para mostrar el uso de este tipo de operación.

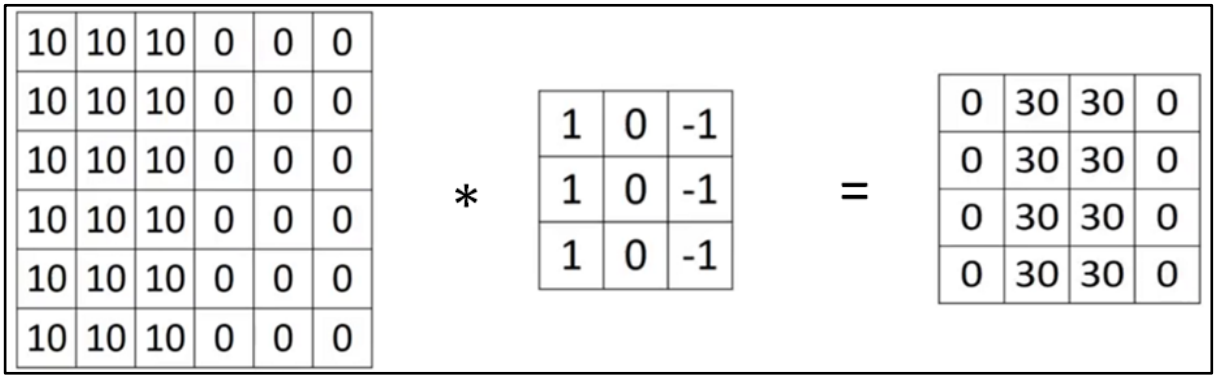

Figura 2.12. Ejemplo de operación de convolución para detección de bordes.

En este ejemplo, se puede observar que el tensor de salida produce elementos que indican un borde detectado en el tensor de entrada. La idea central de una red neuronal convolucional es que el filtro constituye un conjunto de parámetros que se pueden entrenar, por lo que estos modelos resultan ser muy útiles para tareas de computer vision, aunque también se puede emplear este tipo de operaciones para tensores de una dimensión.

Este tipo de redes fueron ideadas por Yann LeCun [18], en donde propone una arquitectura de redes neuronales empleando este tipo de operación. Como se vio recientemente, este tipo de operación resulta ser muy apropiada para la detección de patrones dentro de las imágenes, por ejemplo. Una vez que a través de las diversas convoluciones que se puedan realizar, se ingresa el tensor de salida a redes planas, como las vistas anteriormente, ya que se especializan en la clasificación propiamente dicha. De esta manera se puede establecer un patrón que se sigue para el desarrollo de las arquitecturas de redes neuronales convolucionales. Básicamente se efectúan dos etapas: en la primera se realiza la detección de los patrones a través de la convolución, mientras que en la segunda se clasifica empleando redes planas. En la Figura 2.13 se muestra la arquitectura de la red desarrollada por LeCun, denominada LeNet, empleada para realizar OCR (Optical Character Recognition). Como se puede observar, en dicha figura se emplean un tipo de capas no vistas hasta el momento: las de *pooling*, que sirven para efectuar procesos de reducción de dimensionalidad en los tensores. A continuación, se amplía la explicación sobre el funcionamiento de este tipo de capas.

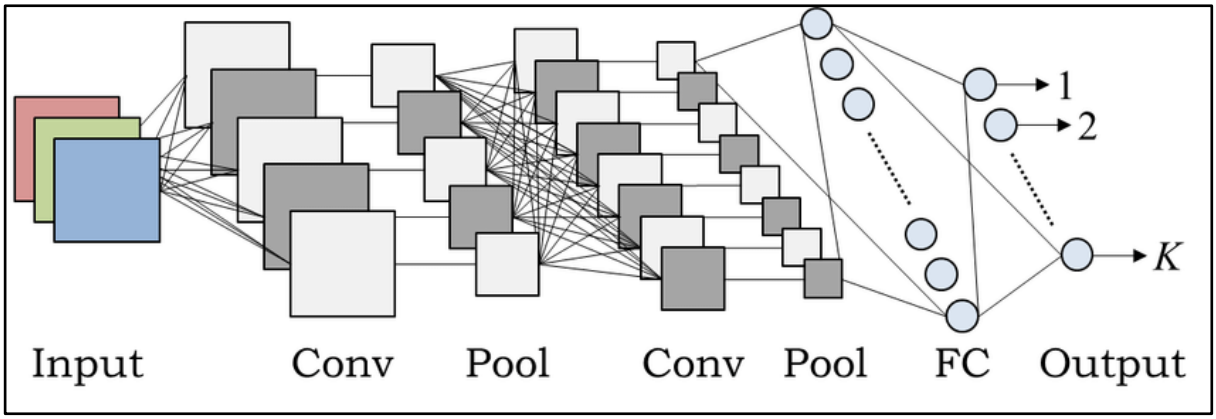

Figura 2.13. Arquitectura de la red neuronal convolucional LeNet.

A partir de lo que se describe en la arquitectura, se puede apreciar que, primero se efectúan las sucesivas convoluciones junto con sus reducciones de dimensionalidad, para luego pasar a las capas Fully Connected (FC), que constituyen otra forma de llamar a las redes planas.

Las capas de pooling se utilizan normalmente luego de efectuar cada convolución a fin de reducir la dimensionalidad del tensor, eliminando elementos obtenidos de la convolución, que posiblemente sean redundantes. Consiste en tomar una "ventana" del tensor, y efectuar una operación fija, que retorne un escalar. Las operaciones fijas que se suelen realizar generalmente es tomar el máximo valor de la ventana, o bien realizar un promedio. En la Figura 14 se muestra un ejemplo de una capa de Max-Pooling, que como indica el nombre, toma el mayor elemento de la ventana.

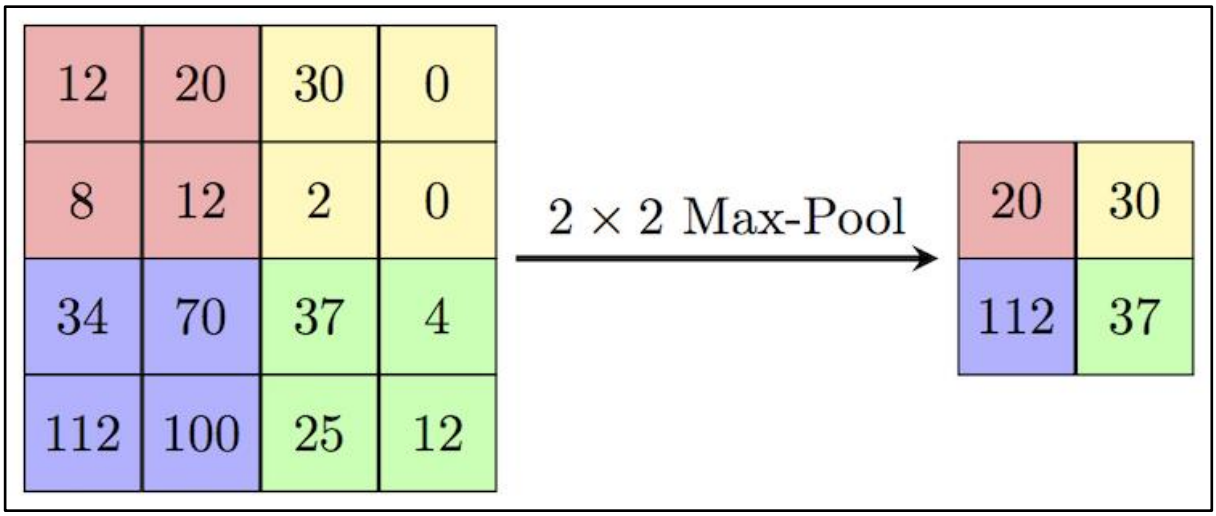

Figura 2.14. Ejemplo de Max-Pooling, que toma una ventana de 2x2, y un stride de 2, es decir que el barrido de la ventana se efectúa moviéndose de a dos posiciones.

El funcionamiento de este tipo de capas es similar al de la convolución en cuanto a que se utiliza una ventana para efectuar un barrido de toda la imagen. La diferencia en este caso es que no hay parámetros, la operación que se realiza es fija.

2.2.3. Redes Neuronales Recurrentes (RNN)

Hasta el momento se mencionaron casos de redes neuronales que funcionan con instancias en donde cada característica se considera independiente de la otra. Es decir, sin que ninguna característica ejerza influencia directa sobre la otra, dentro de la misma instancia. Mientras que las redes convolucionales son típicamente empleadas para tareas de Computer Vision (CV), las redes recurrentes es muy común que sean empleadas para tareas de Natural Language Processing (NLP). Un problema típico de NLP que se resuelve con RNN es, por ejemplo, identificar el significado de

una oración con una determinada disposición de las palabras, o el significado de la misma oración con el mismo conjunto de palabras en otro orden y con otro significado.

Las redes recurrentes se caracterizan por consistir en unidades cuya salida vuelve introducirse en la celda. Es decir que la activación de la primera feature de la instancia, puede influir en la activación de la segunda feature, y así sucesivamente para todas las features de la instancia a someter a la red. De esta manera, estas redes se diferencian de las detalladas anteriormente, ya que ni las planas ni las convolucionales forman ciclos, por lo que se las suele llamar como redes *feedforward*.

El entrenamiento de este tipo de capas es muy costoso computacionalmente, provocando que el proceso de entrenamiento con CPU sea prácticamente inviable. En la Figura 2.15 se muestra un ejemplo gráfico de la apariencia que tienen las celdas recurrentes.

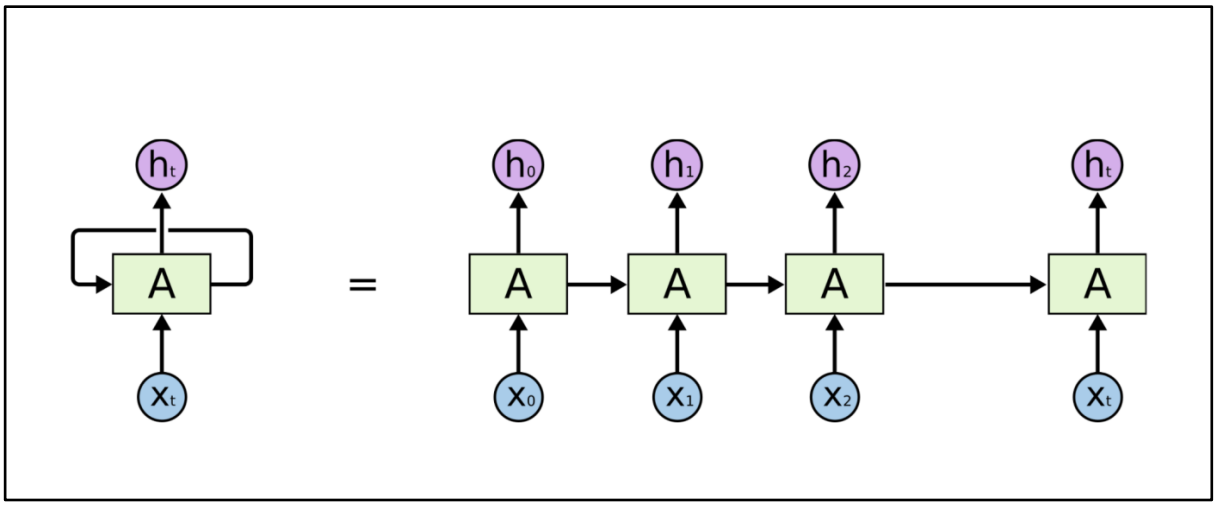

Figura 2.15. Ejemplo de celda recurrente, en donde la activación de una instancia tiene tantos ciclos como features tenga, lo cual provoca que el entrenamiento de las mismas sea sumamente costoso.

El primer trabajo realizado sobre este tipo de problemas fue por Hochreiter et. al en 1997 [19], con un tipo de unidad llamada Long Short-Term Memory (LSTM). En donde dentro de cada celda se efectúan varios cálculos para así lograr un efecto de "memoria", el cual en función del valor que posea dicha memoria, se pondera la activación del ciclo siguiente, por decirlo de forma rudimentaria. La Figura 2.16 muestra cómo se ve una celda LSTM, en donde se podrá observar que hay varias entradas y salidas.

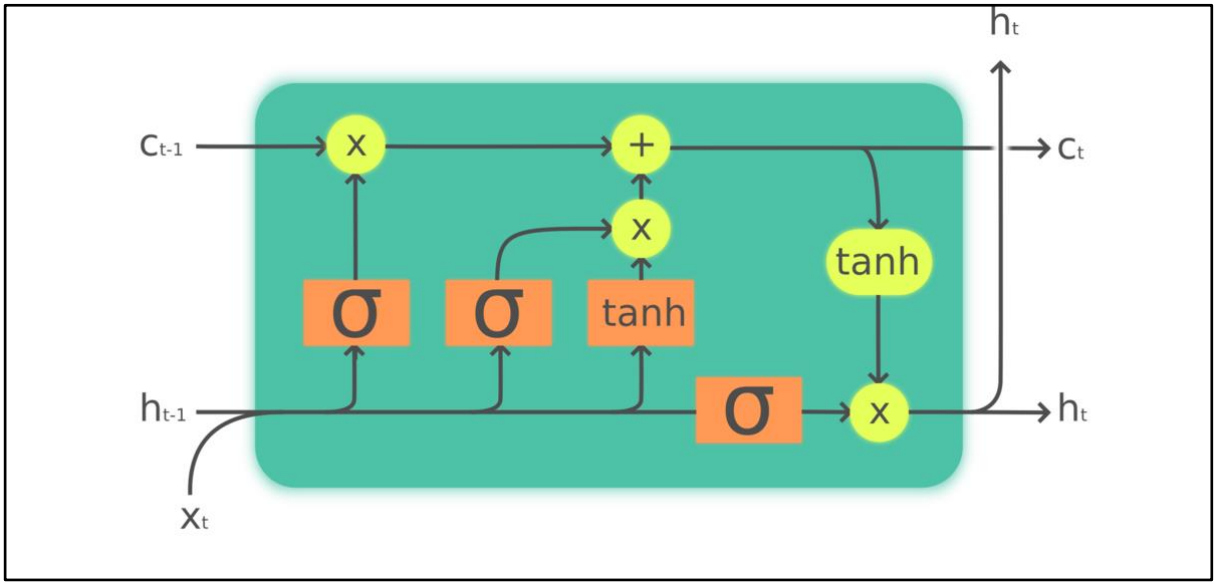

Figura 2.16. Celda LSTM. Se tienen dos salidas en este caso, la activación correspondiente y el valor de memoria, ingresando ambos al próximo ciclo de la recurrencia. Fuente: Wikipedia[6]

Como se puede observar, a modo de palpar la complejidad que tienen las redes recurrentes, se tiene que al pasar por una capa LSTM, una instancia de por ejemplo 300 features, debe efectuar 300 ciclos, es decir uno por cada feature. Para cada ciclo, deben realizarse todos los cálculos que se muestran en la imagen. Por este motivo, para entrenar una red neuronal basada en LSTM, se necesita definitivamente emplear una GPU. Sin embargo, la capacidad de aprendizaje de este tipo de redes es superior, siempre y cuando el dataset contenga la información adecuada.

Luego existen otro tipo de celdas que se emplean para las redes recurrentes, que son las Gated Recurrent Unit (GRU), desarrolladas por Cho et al. [20] en donde se proponen celdas que son más simples que las LSTM, es decir que se efectúan menos cálculos dentro de una GRU. Esto provoca que aprendan menos que las LSTM, pero que sean más "escalables", es decir que, para redes más grandes, las LSTM al ser muy complejas se quedan atrás al provocar que el entrenamiento sea sumamente lento. Como contrapartida, las GRU en esos casos llevan la ventaja al entrenar más rápido. Generalmente lo que se recomienda es que a menos que la arquitectura de la red neuronal sea muy grande, usar siempre por defecto LSTM. A fin de comparar la diferencia entre cada tipo de unidad en cuanto a complejidad, se muestra en la Figura 2.17 la apariencia que tiene una GRU.

[6] https://en.wikipedia.org/wiki/Long_short-term_memory

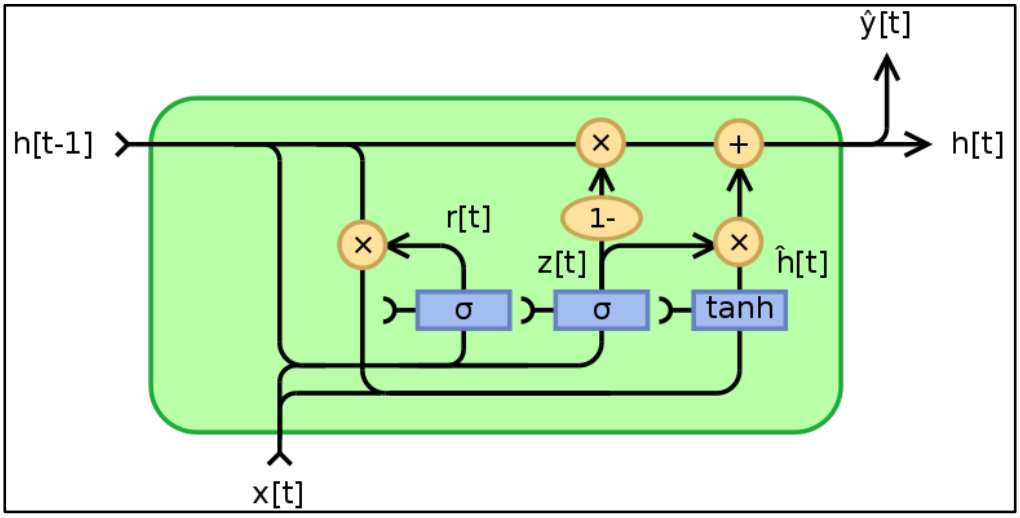

Figura 2.17. Celda GRU en donde a diferencia de LSTM, existe una función de activación menos. Fuente: Wikipedia[7]

Con respecto a las arquitecturas que tienen las redes neuronales recurrentes, siguen una disposición similar a la de las redes convolucionales, en donde se presentan dos etapas. La primera para la detección de patrones para una instancia, y la segunda para la clasificación propiamente dicha. En la primera etapa es donde lógicamente se emplean las capas recurrentes, mientras que en la segunda se efectúa la clasificación utilizando las redes planas, es decir empleando redes densas.

2.2.4. Trabajos Relacionados

Una vez introducidos los aspectos esenciales de las redes neuronales, se pueden comprender los trabajos relacionados a estos conceptos, los cuales fueron empleados como motivación para el desarrollo de los modelos predictivos de este trabajo de grado. Afortunadamente hay mucho trabajo al respecto, ya que al ser Deep Learning uno de los temas del momento, es mucha la cantidad de gente que está trabajando en tema, tanto de la industria como de la academia. De esta manera, se citarán algunos de los trabajos más relevantes de los últimos años.

Antes de abordar los trabajos relacionados a técnicas más aplicadas a las arquitecturas de las redes neuronales, es muy importantes destacar trabajos muy importantes que también se relacionan al área, más precisamente con la representación de la información. Es conocido que, en un problema de aprendizaje automático, un gran paso es el preprocesamiento del dataset. Generalmente este paso es el más importante de todo el proceso predictivo, ya que una representación lo suficientemente depurada, que hace que los modelos se puedan ajustar bien, es más importante que el desarrollo del modelo en sí. Sin embargo, el preprocesamiento de los datos suele ser una tarea tediosa, por lo que no es raro observar que no se dedique el tiempo necesario a esta etapa del proceso.

[7] https://en.wikipedia.org/wiki/Gated_recurrent_unit

A partir de 2013, Tomas Mikolov et al. iniciaron una línea de investigación enfocada a la representación del texto [21, 22, 23]. En dicho entonces, si bien existían métodos para representar texto, el alcance y las posibilidades que permitían eran mínimos. Se idearon entonces los word-embeddings, que consisten en el resultado de aprender representaciones vectoriales de las palabras. El resultado fue que cada palabra se mapea a un vector de longitud fija, en donde cada elemento contiene un valor real que indica la intensidad con la que se relaciona la palabra original, con un determinado concepto. Por ejemplo, la palabra "rey", se mapea a un vector en donde el elemento i, que se relaciona con el concepto "género", posee un valor de -1, mientras que la palabra "reina", se mapea a un vector cuyo elemento en la posición i, tendrá un valor de 1. En la Figura 2.18 se muestra un ejemplo gráfico de cómo funcionan los embeddings. Además, esta noción no sólo se aplica a palabras, sino que se pudo extender a oraciones, resultando en embeddings de longitud fija, representando a una oración. Luego con respecto a este tipo de representación de información, existen otros trabajos que añaden información morfológica a los embeddings [24], en donde se obtienen resultados muy satisfactorios.

Cabe destacar que los word-embeddings se entrenan en función a un corpus de texto, por lo que la relación que una palabra tiene con un determinado concepto, estará atada a dicho corpus. Debido a esta cuestión, es de vital importancia que el corpus sea lo más grande y variado posible, para así poder generar embeddings confiables, aptos para que puedan ser empleados en otros contextos. Con respecto al entrenamiento de los mismos, el mismo se lleva a cabo mediante diversos algoritmos, como por ejemplo Word2Vec [22] o FastText [25]. En ambos algoritmos, se tienen diversos hiperparámetros que definen ciertas cuestiones como por ejemplo la cantidad de dimensiones de los embeddings. Si bien se puede elegir una dimensión más grande para abarcar más conceptos, a mayor dimensión se puede perder eficacia al momento de entrenar. Por otro lado, si se establece una dimensión pequeña para los embeddings, los mismos serán precisos, pero abarcarán pocos conceptos.

	Man	Woman	King	Queen	Apple	Orange
Gender	−1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.70	0.69	0.03	-0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97

Figura 2.18. Ejemplo de word-embeddings. En este caso se tratan de vectores de longitud fija de 4 elementos, en donde cada uno representa un concepto que se indica con un color distinto.

Con respecto a técnicas que trabajan directo sobre el modelo predictivo, se pueden mencionar algunos trabajos que han sido de gran relevancia e influencia para el área de aprendizaje automático y procesamiento de lenguaje natural.

El parsing de dependencias dentro de un texto, es una rama muy vigente. Por ejemplo, Kiperwasser et al. [26] presentaron un esquema simple y efectivo para efectuar dicha tarea. Este modelo se basa en redes neuronales recurrentes bidireccionales, más precisamente en capas LSTM bidireccionales (Bi-LSTM). Dentro de las redes neuronales recurrentes, se mencionó anteriormente que la activación en una celda, es reingresada en la próxima activación, es decir en el siguiente ciclo. De esta manera se forma una recurrencia hacia adelante, iniciando con la primera feature de la instancia, para luego finalizar con la última. En una red bidireccional, se efectúa una tarea más, que es realizar los ciclos recurrentes, pero también comenzando desde la última feature, para finalizar con la primera. De esta manera se establece un efecto de memoria tanto para adelante como hacia atrás (Figura 2.19). En este trabajo los resultados que se obtuvieron establecieron un nuevo estado del arte en parsing de dependencias.

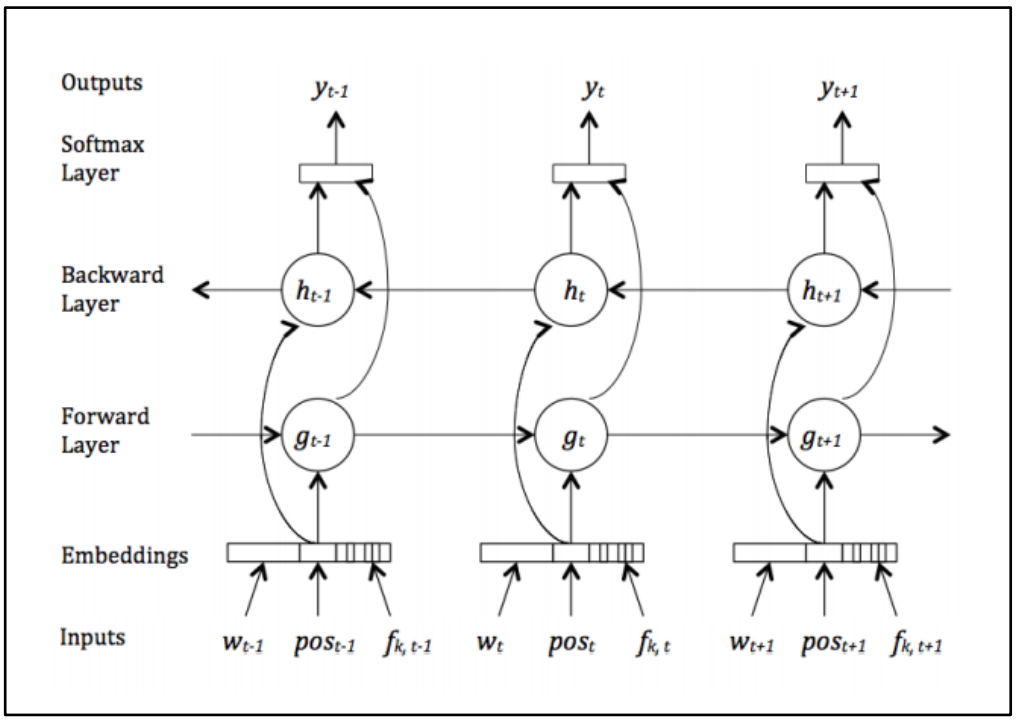

Figura 2.19. Ejemplo de una red con una capa recurrente bidireccional.

Con respecto a redes convolucionales, también se han realizado diversos trabajos sobre NLP, con resultados exitosos. En 2014 Y. Kim desarrolló un modelo basado en redes neuronales convolucionales a partir de word-embeddings [27], para realizar tareas de clasificación de documentos y análisis de sentimiento. Luego Zhang et. al [28] propusieron realizar clasificación de texto a partir de la consideración de los caracteres, de forma individual. Otro caso exitoso de aplicación de redes convolucionales consistió en la aplicación de redes convolucionales para la detección de sarcasmo en medios sociales [29, 30].

Luego existen otros casos en donde se usan ambos enfoques. Es decir que emplean modelos con capas convolucionales y recurrentes. Por ejemplo, el sistema de reconocimiento de voz de Microsoft [31] publicado en 2017, en donde utiliza como primera etapa capas convolucionales, para luego utilizar capas Bi-LSTM que actúan sobre la selección de features realizada por las convoluciones. Por último, utilizan capas densas para efectuar la clasificación final.

Por último, se hace referencia a modelos que traducen audio a caracteres. Uno de los trabajos más relevantes en este tema es Listen, Attend and Spell (LAS) [32]. Este modelo consiste en lo que los autores denominan Bi-LSTM piramidal, en donde la cantidad de unidades que contienen las sucesivas capas Bi-LSTM se va reduciendo. Luego utilizan otras capas recurrentes típicas para obtener el caracter final.

En la teoría relacionada, se vio que el método que comúnmente se emplea es el gradiente descendente. En donde a partir de una función de costo, se computa el gradiente para obtener la dirección de máximo decrecimiento, a fin de optimizar los parámetros tal que minimicen el costo. Sin embargo, no es el único optimizador que se emplea, en la práctica existen otros métodos que, si bien también están basados en el gradiente descendente, realizan una serie de ajustes en la parte matemática a fin de obtener varias ventajas, como por ejemplo obtener errores menores, convergencia más rápida, etc.

Entre estos métodos se encuentra Adam, un algoritmo de optimización de primer orden que se basa en momentos adaptativos [33]. Se trata de un método computacionalmente eficiente y que tiene bajos requerimientos de memoria. Es uno de los que mejor resultados suelen dar al momento de entrenar las redes, sobre todo en caso de redes basadas en LSTM. A partir de este método se encuentra AdaMax, una variante del mismo que varía en la forma en que pondera el gradiente. Según se indica en la publicación, el algoritmo es muy estable, provocando que sea mucho más robusto frente a posibles ruidos en los gradientes, en comparación con Adam.

Otro de los optimizadores que suele emplearse, que generalmente es con el cual se obtienen las mejores métricas, es Adadelta [34]. Se trata de un optimizador que dinámicamente se adapta en el tiempo empleando información de primer orden, con mínimo overhead respecto al gradiente descendente corriente. El método se caracteriza por no requerir ajuste manual del learning rate, y por mostrarse también robusto al problema del ruido en la información del gradiente. Dicho algoritmo es comúnmente comparado con Adagrad [35], otro optimizador de similar funcionamiento. Este tipo de algoritmos de gradiente adaptativo suelen ser las opciones más escogidas al momento de trabajar con redes neuronales.

Con respecto a otras técnicas importantes que se suelen utilizar en las redes neuronales, se destacan algunas relacionadas al tratamiento del overfitting, un problema que aparece también en problemas de aprendizaje automático. Si bien un modelo lo que hace es ajustarse al conjunto de datos suministrado, en caso de que un modelo se ajuste demasiado a dicho dataset, se perderá accuracy debido a la pobre capacidad de generalización. Este fenómeno se denomina *overfitting*, el cual es muy común en este tipo de problemas, y constituye ser una situación altamente indeseable al momento de desarrollar modelos predictivos. En la Figura 2.20 se provee un ejemplo de overfitting, en donde se puede apreciar que el modelo que ajusta con la línea negra obtiene mejores resultados

debido a su mejor capacidad de generalización, en comparación con el modelo identificado con la línea verde.

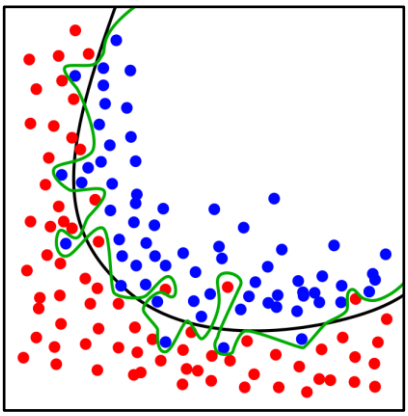

Figura 2.20. Ejemplo de overfitting en donde el modelo identificado con verde está sobre ajustado al dataset.

Una técnica comúnmente utilizada para combatir el problema del overfitting es Dropout [36], que consiste en la desactivación de la salida de las neuronas, a partir de un valor probabilístico parametrizado. Para cada unidad de una determinada capa, se obtiene un valor aleatorio, y si dicho valor sobrepasa al valor probabilístico establecido como parámetro, la salida de dicha unidad es establecida en cero. El valor establecido como umbral (llamado keep_rate) constituye un hiperparámetro más de la red neuronal.

Otra técnica que suele ser utilizada para regularizar es Batch Normalization [37], que consiste en distribuir los datos a fin de facilitar y acelerar la activación. En este paso es importante destacar que, si bien la normalización es comúnmente utilizada para efectuar regularización sobre el modelo, no es la intención que tiene la normalización. Este proceso añade un poco de ruido de las activaciones de las capas ocultas en cada mini-batch, lo cual es verdad que provoca un pequeño efecto de regularización, pero puede derivar en efectos indeseados, siendo muy perjudicial para el proceso de aprendizaje.

2.3. Conclusión

En este capítulo se efectuó un barrido por las bases teóricas sobre las cuales se apoya el desarrollo del presente trabajo de grado, junto con un conjunto de trabajos relacionados a cada tema. Dichas bases teóricas giran en torno a dos ejes. Se realizó especial énfasis en el modelado de usuarios, abordando en profundidad los modelos teóricos empleados: IPA y Symlog. Para dichos modelos se mostró en profundidad la forma en la que trabajan, ofreciendo diversos ejemplos abarcando todos los detalles inherentes a cada marco. De esta manera se dejan firmemente asentadas las bases que explican la semántica y el contexto en cual se da el desarrollo del modelo predictivo basado en redes neuronales.

Con respecto a las técnicas de aprendizaje profundo, se comenzó detallando los conceptos más básicos, como por ejemplo la regresión lineal, para finalmente haber explicado el funcionamiento de las redes neuronales recurrentes. Entre medio se abordaron los aspectos más importantes de aprendizaje profundo, a saber: redes neuronales planas, redes neuronales convolucionales, recurrentes, y todos los detalles de las arquitecturas de redes basadas en estos conceptos. De esta manera se dispusieron las herramientas necesarias para poder desarrollar exitosamente un modelo predictivo de estas características. Partiendo de la base que el dominio sobre el cual se aplican estas técnicas es el modelado de usuarios, el sesgo que se produce en la explicación de las redes neuronales se ubicó más en temas relacionados a Natural Language Processing (NLP), que a Computer Vision (CV), que es también un tema muy relacionado con estas técnicas.

Capítulo 3: Desarrollo de la Propuesta

En este capítulo se desarrolla la propuesta que motiva al presente trabajo, entrando en detalle sobre las decisiones tomadas que resultaron en la herramienta presentada. La solución empleada consiste de varios aspectos, comenzando por la visión general de la herramienta desarrollada. Luego se descienden varios niveles de abstracción en las sucesivas secciones para explicar la estructura del modelo predictivo, ya que, a nivel de arquitectura de la aplicación, dicho modelo puede ser tratado como una caja negra.

En la Sección 3.1. se abordan los detalles de la herramienta presentada, cuya explicación se divide, a grandes rasgos, en un apartado dedicado al cliente, y otro al servidor. Se comienza explicando la propuesta desde la perspectiva de una aplicación web, para luego explicar la aplicación que corre en el backend, en donde se realiza especial énfasis. También se detalla la forma en que se generan los perfiles, manteniendo al clasificador como una caja negra, para luego profundizar en las sucesivas secciones.

En la segunda sección se desciende un nivel en la abstracción de la explicación, para poder explicar la organización del modelo predictivo. Se muestra que el clasificador basado en técnicas de aprendizaje profundo está organizado en dos etapas, las cuales fueron dispuestas de tal manera que reduzca la dificultad en la clasificación. También se detalla la forma en que se generan los perfiles, aunque aún sin interiorizarse en la teoría subyacente del clasificador, pero ya explicando a un nivel más fino que en la sección anterior. Las cuestiones teóricas de las técnicas de aprendizaje profundo se reservan para la última sección.

Por último, en la Sección 3.3. se desciende un nivel más en la abstracción del clasificador. Se comienza explicando la representación de la información, la cual está basada en el concepto de word-embeddings. Luego a partir de dicha representación, se comentan las técnicas empleadas en cada enfoque de clasificación, es decir la teoría de redes neuronales que se emplea en cada enfoque propuesto.

3.1. Estructura de la Herramienta

La herramienta propuesta consiste en una aplicación en la que, a partir de un conjunto de mensajes provistos por un usuario, se genera una serie de indicadores asociados a cada usuario. Esto quiere decir que los mensajes deben estar asociados a determinados datos, que son: el remitente del mensaje, el grupo de trabajo de dicho individuo, y la marca de tiempo en donde se envió dicho mensaje. En esta sección no se hace mención a la forma en que se generan los indicadores, ya que se abordan en detalle en las secciones 3.2. y 3.3.

Los indicadores que se muestran a través de la aplicación, constituyen los perfiles de usuario, por lo que cada persona tendrá un determinado perfil de usuario. Además, puede ocurrir que, en el conjunto de mensajes provisto, un determinado individuo haya estado presente en más de un grupo de trabajo. En ese caso, toma relevancia la marca de tiempo, ya que provee la posibilidad de monitorear la evolución de los perfiles de usuario de una determinada persona, a lo largo del tiempo.

A continuación, se detalla la estructura de la herramienta, a partir de la cual se comienza explicando la estructura de la misma desde la perspectiva del cliente, para luego enfocar la explicación desde el lado del backend, a fin de seguir la línea de comenzar por lo más abstracto para luego ir descendiendo progresivamente.

3.1.1. Front-End

La funcionalidad descrita recientemente se lleva a cabo a través de una aplicación web. Dicha aplicación se encarga básicamente de darle un tratamiento especial a la salida que ofrece el backend, ya que una de las premisas principales es que el usuario final no tenga que provenir necesariamente del área de sistemas. De esta manera, informarle al cliente los perfiles de usuario a partir de un archivo JSON no sería la decisión más conveniente en términos de usabilidad.

A partir de dicha premisa, se desarrolla un front-end encargado de darle un formato a la información, con una interfaz como la que se expone en la Figura 3.1. La idea de emplear una aplicación web recae en que el procesamiento de los archivos no se debe realizar en el equipo del cliente, sino de forma remota. De esta manera, el usuario final tampoco necesitará realizar ninguna instalación, por lo que desaparece la cuestión relacionada a instalar tecnologías específicas para garantizar el funcionamiento de la aplicación. Solo bastará el uso de un navegador en donde se deba acceder a la URL correspondiente a la aplicación web.

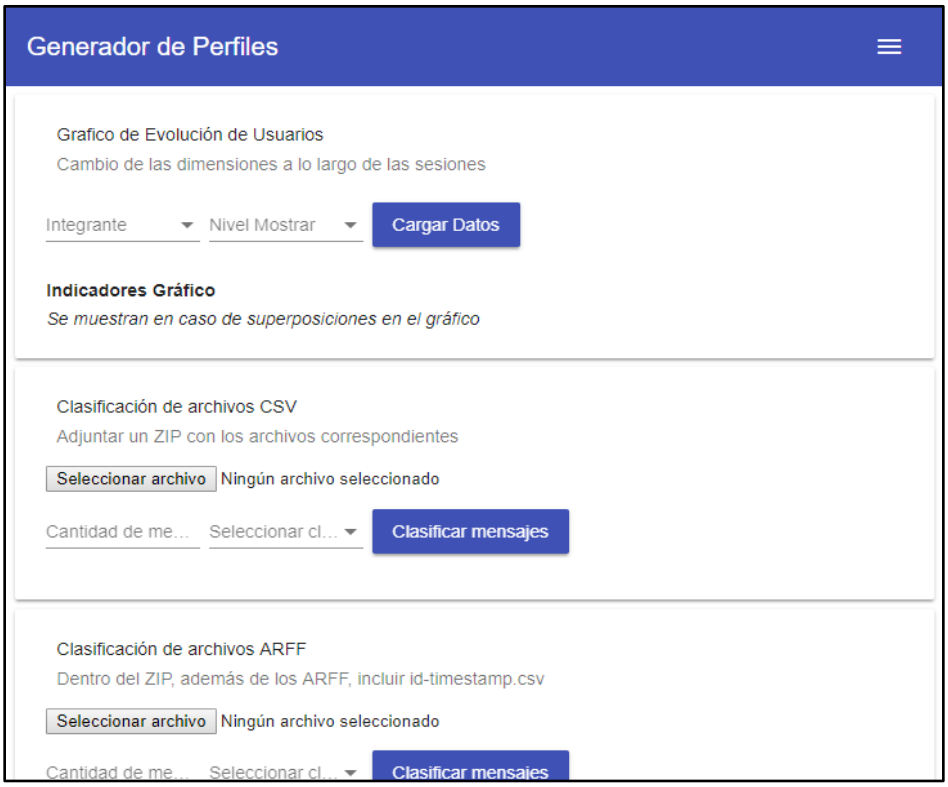

Figura 3.1. Pantalla de la aplicación en donde se observa el aspecto de la interfaz provista.

Como se puede observar, el usuario final podrá subir sus fuentes de información, en determinados formatos soportados. Estos son: ARFF, JSON, CSV, y a partir de una base de datos (proporcionando los datos de conexión a la misma). Una vez que el usuario termine de cargar dichos datos, y de enviarlos, deberá esperar un tiempo proporcional a la cantidad de datos que envió, para poder obtener una respuesta del backend. Una vez que se obtenga la respuesta, la misma será mostrada de la siguiente manera: se dispondrá una tabla en donde cada fila se corresponderá con un usuario y cada columna se corresponderá con cada uno de los indicadores. Por razones de simplicidad, los indicadores se separan en grupos que se muestran en tablas separadas, a fin de facilitar la visualización. Por ejemplo, las conductas y las reacciones van en tablas separadas.

A modo de ejemplo, se ofrece la Figura 3.2, en donde se muestra una tabla con uno de los grupos de indicadores que se muestran en la aplicación web. En dicha figura, se muestra el grupo de indicadores correspondientes a las reacciones de IPA. Los grupos de indicadores que se corresponden con la teoría IPA y Symlog, haciendo el foco en esta última, por lo que los grupos de indicadores que constituyen cada perfil de usuario son: conductas, reacciones, indicadores Symlog, y dimensiones Symlog.

Figura 3.2. Tabla con la visualización del grupo de indicadores correspondiente a las reacciones IPA.

Luego, a partir de las marcas de tiempo que cada grupo de trabajo tiene asociado, se generan gráficos de líneas de dos dimensiones, ordenados por tiempo. Se muestran los mismos indicadores que en las tablas, pero separados por cada grupo de trabajo, ordenados por tiempo. Los parámetros de los gráficos son, el grupo de indicadores a mostrar (por ejemplo, reacciones o indicadores Symlog), y el integrante que se quiere monitorear. A modo de ejemplo, se muestra en la Figura 3.3 un gráfico asociado a la Figura 3.2, que muestra las reacciones para un determinado individuo a lo largo de todas las sesiones en las que estuvo presente.

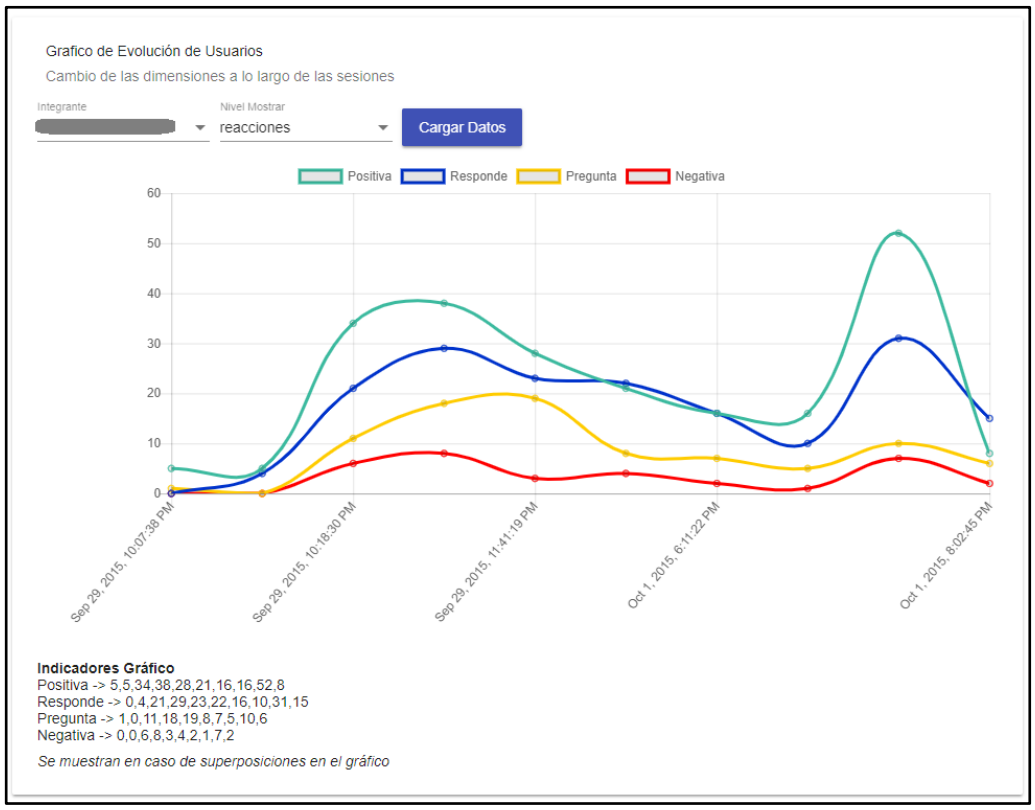

Figura 3.3. Gráfico que muestra las reacciones de la tabla para un determinado integrante, pero separando por sesión de trabajo.

Otra funcionalidad que se incluirá, a fin de optimizar la usabilidad de la aplicación web, es la posibilidad de que el usuario final pueda consultar clasificaciones ya realizadas. De esta manera, para obtener una determinada clasificación, no hace falta clasificar por cada momento que se necesiten los perfiles, siempre y cuando la predicción se haya realizado al menos una sola vez. Para esto último, se provee también la posibilidad de que los resultados se puedan persistir en una base de datos, a fin de no tener que recurrir a ningún tipo de cookie para almacenar los resultados en la terminal del usuario final. Como se podrá observar, la idea de que toda la funcionalidad esté en la nube se tiene en cuenta en todo momento, a fin de ofrecer una experiencia de usuario que no requiera ataduras de uso a un equipo en particular.

3.1.2. Back-End

El backend es el encargado de que, a partir de un archivo que contenga las interacciones, es decir lo mensajes, devuelva los indicadores asociados a cada individuo en cada sesión de trabajo. Para poder llevar a cabo dicha tarea, se necesita una determinada cantidad de información, como bien se introdujo en la presente sección.

La idea central es que, sin importar la naturaleza de los mensajes ni el contexto, se puedan obtener los perfiles asociados. Para ello, se imponen ciertas restricciones en cuanto al formato y a la información que se provee. El eje de la cuestión es solicitar la mínima información necesaria, sin que suponga un trámite engorroso para el usuario, lo cual recae en la necesidad de que se proporcione lo siguiente para cada mensaje:

- Usuario que envió el mensaje
- Mensaje concreto enviado por el usuario
- Grupo de trabajo al cual pertenece
- Marca de tiempo en la cual se envía el mensaje

La necesidad por la cual se necesita incluir el usuario que envía el mensaje, se puede deducir sencillamente: para asociar los mensajes con el remitente. Luego surgen dos datos más que sirven para efectuar el análisis que se va a realizar sobre los mensajes clasificados. El grupo de trabajo se solicita con el objetivo de agrupar las clasificaciones por cada grupo de trabajo en el cual participa cada usuario. Este concepto carecería de relevancia en casos donde cada usuario haya participado en un solo grupo de trabajo. Aunque para casos en donde el mismo usuario haya participado en varios grupos de trabajo, se puede monitorear la variación en el comportamiento de un usuario a lo largo de cada sesión, como bien indica la Figura 3.3. Esto le permitirá al usuario final que esté observando dicha información, tomar las acciones pertinentes en caso de encontrarse con desvíos inesperados en los indicadores de alguna sesión.

Luego se solicita la marca de tiempo asociada a cada mensaje. De igual manera que el grupo de trabajo, en casos donde cada usuario haya pertenecido a solo un grupo de trabajo, este tipo de información carecería de relevancia. Esta cuestión se ve modificada en gran medida cuando un usuario integró varios grupos de trabajo, ya que se puede ordenar los comportamientos en cada sesión por orden de tiempo. Dicho orden permite la posibilidad de monitorear la evolución (o involución) del comportamiento de un individuo a lo largo del tiempo. En caso de no contarse con dicho dato, las distintas conversaciones no podrían ordenarse de tal manera, lo cual imposibilita monitorear las variaciones en el comportamiento de los individuos.

A modo de resumen, cada mensaje que el usuario final provea, debe tener una estructura como la que se ejemplifica a continuación en la Figura 3.4. Como se podrá observar en la imagen, el ejemplo se corresponde con un solo mensaje. El conjunto de mensajes a proveer, consiste en un listado de dicho formato de mensajes.

Figura 3.4. Tipo de formato de mensaje con el cual trabajará el framework. Como se puede observar, el tratamiento informático de dicho formato se da de forma sencilla mediante un archivo CSV.

El formato que se expone es el utilizado internamente por el framework, por lo tanto, es el único que entiende. Esto quiere decir que el usuario final debe proporcionar su conjunto de mensajes en este formato. Sin embargo, a fin de facilitar la tarea, se debe poder trabajar con distintas fuentes de información, por lo que se da soporte a otros formatos como por ejemplo el Takeout[8], que consiste en los backups de los mensajes enviados a través de los servicios digitales de Google. Si bien el formato de estos backups es un JSON, el backend cuenta con mecanismos para efectuar la traducción de JSON a CSV para poder manejar la información.

Otro formato comúnmente utilizado para tareas de clasificación con técnicas de aprendizaje automático es el ARFF, utilizado para clasificadores de la herramienta WEKA. De igual manera que los Takeout, el framework desarrollado incluye mecanismos de traducción de dicho formato a CSV, a fin de compatibilizarlo con las redes neuronales empleadas, que trabajan con CSV. De esta manera, se le solicita al usuario que, junto con los ARFF, incluya un archivo CSV simple que asocie un identificador de conversación, con la marca de tiempo donde se desarrolla la conversación. Con respecto al identificador de la conversación, el framework tomará dicha información a partir del nombre del archivo ARFF. En este caso se supone que cada archivo ARFF se corresponde con una conversación diferente.

Por último, el backend desarrollado prevé la posibilidad de considerar que el usuario final no necesite proveer los mensajes a través de archivos, sino a través de una base de datos. Este tipo de posibilidad se puede dar a partir de los datos de conexión con la misma, y a partir de un formato específico de base de datos. De esta manera el framework accederá a la base de datos provista y extraerá los mensajes; la conversión a CSV se dará automáticamente. En el Capítulo 4, relacionado a la implementación, se proveen más detalles del formato de persistencia soportado.

3.2. Proceso Predictivo

Hasta el momento se habló que, al momento de efectuar la clasificación a partir de los mensajes, se obtenían una serie de indicadores que constituyen los perfiles de usuarios. Sin embargo, no se realizó mención alguna a cómo se generan dichos indicadores.

Se sabe que los indicadores son generados a partir de un conjunto de mensajes que el usuario proporciona, junto con un determinado tipo de información. Pero al momento de la predicción, lo que se clasifica concretamente son los mensajes, es decir el contenido textual de cada mensaje. La salida con la cual se desarrollaron los clasificadores, no son los indicadores concretos, sino conductas IPA. Esto quiere decir que el proceso predictivo no se reduce solamente al uso de clasificadores, sino que se tienen otros aspectos como por ejemplo una propuesta de correspondencias, que mapean cada reacción con un determinado indicador Symlog.

Como las reacciones constituyen una agrupación de las conductas IPA, posibilitan mejoras en la clasificación de los mensajes al poder clasificar por etapas. Todos estos aspectos se abordan con

[8] https://en.wikipedia.org/wiki/Google_Takeout

detalle en la presente sección. Cabe destacar que si bien en la Subsección 3.2.2. se explica la estructura de clasificación, las técnicas concretas de Deep Learning se comentan en la Sección 3.3.

3.2.1. Pipeline

El punto de partida para generar los perfiles de usuario son los mensajes que el usuario proporciona a la aplicación web. Para la clasificación de los mensajes se comienza con la teoría IPA, que básicamente consiste en doce categorías de conductas. Como cada mensaje se clasifica con una determinada conducta, el problema se enmarca en un típico problema de clasificación de documentos, en donde la entrada es un determinado texto, y la salida es una categoría. En este caso, la categoría puede ir del 1 al 12, según establece IPA.

A partir de dicha salida, se efectúa una propuesta de mapeo a Symlog, a partir de las conductas. De esta manera el proceso de generación de perfiles queda cerrado e ilustrado en la Figura 3.5, en donde a partir del texto plano provisto por el usuario (mensajes), se obtiene la categorización para finalmente obtener los valores Symlog. Como se podrá observar en la figura, las técnicas de aprendizaje profundo se aplican sólo para la obtención de las conductas asociadas a cada mensaje.

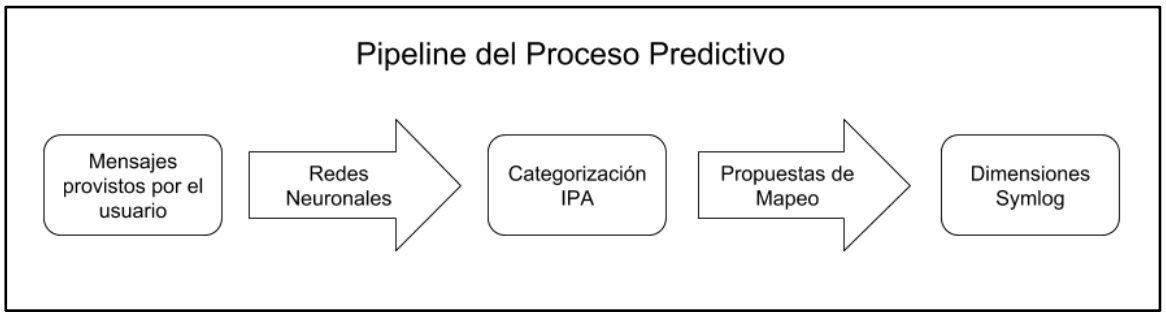

Figura 3.5. Ilustración abstracta del proceso predictivo que se lleva a cabo. Se puede observar que las fuentes de información que provee el usuario, es clasificada a nivel mensaje, para luego trabajar sobre dichas clasificaciones, resultando en dimensiones Symlog sobre el final.

A partir de la conducta predicha por el clasificador, se obtiene la reacción asociada. Por ejemplo (Takeout), si la red neuronal predijo la conducta C3, la reacción asociada es “Positiva”. De esta manera, las conductas predichas para los mensajes, se mapean a su correspondiente reacción, para luego poder armar los indicadores Symlog. En la Figura 3.6 se muestra el mapeo completo que resulta en cada uno de los seis indicadores Symlog.

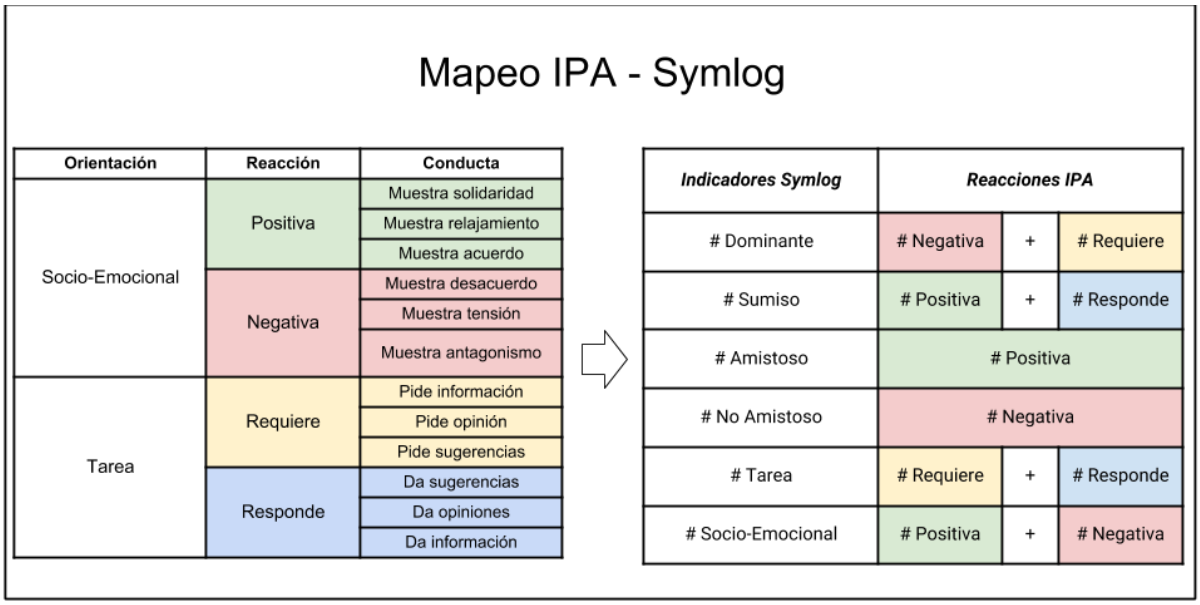

Figura 3.6. Obtención de los indicadores Symlog a partir de las conductas IPA [38]. A partir de las conductas del 1 a 12 que se obtienen del clasificador (tabla izquierda), se agrupan en reacciones. Esas reacciones son las que luego se utilizan para armar los indicadores (tabla derecha).

Como se podrá observar, los datos con los que finalmente cuenta el supervisor, no constituyen solamente los de las conductas predichas por las redes neuronales, sino también los indicadores Symlog. Estos indicadores se establecen a partir de las predicciones de conductas. De esta manera, el resultado para las fuentes de información que el usuario final suba, son una serie de indicadores asociados a cada individuo presente. Se tienen doce valores, correspondientes a las conductas y seis indicadores de Symlog. Cabe destacar también que estos indicadores son para cada persona, pero no de forma general. Es decir, que dichos indicadores serán retornados por el framework para cada persona, pero por cada sesión. De esta manera se posibilita la construcción de diversas métricas para el monitoreo de la evolución de la personalidad de cada usuario, sea a través de tiempo, en función al resto de los integrantes, etc.

3.2.2. Estructura del Clasificador

La teoría IPA, si bien determina que son 12 las conductas posibles para cada mensaje, permite que se puedan agrupar en 4 reacciones. De esta manera, para cada reacción, se tienen tres conductas posibles. Por más que se comente esta agrupación en reiteradas ocasiones esta idea indica que puede ser aprovechada para el desarrollo de los clasificadores, permitiendo un proceso de clasificación en dos etapas. Una primer etapa se clasifica el mensaje en una de las cuatro posibles reacciones, mientras que la segunda etapa, partiendo de la reacción predicha, se clasifica el mensaje en una de las tres posibles conductas pertenecientes a dicha reacción. Esta técnica divide el problema de clasificar en 12 categorías diferentes en un problema de clasificar en 4 categorías y luego en 3. La Figura 3.7 grafica el proceso de clasificación en etapas que se lleva a cabo. En dicha imagen se puede observar que, el problema pasó de necesitar un solo clasificador para predecir

doce conductas, a cinco clasificadores, pero con menos categorías a predecir. Dicha idea constituye un elemento clave para mitigar la probabilidad de error al momento de elaborar predicciones.

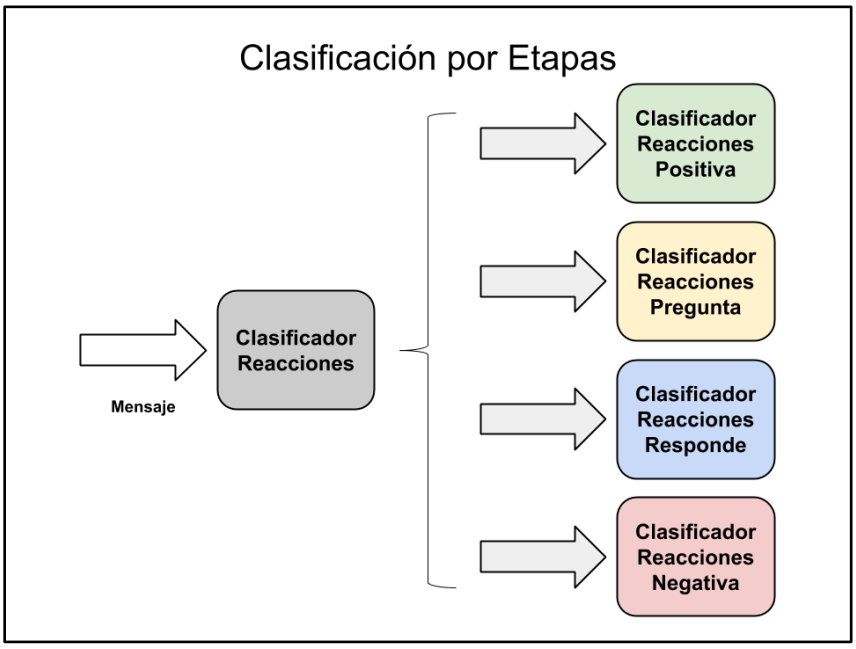

Figura 3.7. El mensaje es sometido a un clasificador de reacciones. A partir de la reacción predicha, el mismo mensaje es clasificado por el segundo clasificador que le corresponde.

3.3. Enfoques Propuestos

En esta sección se tratan aspectos más concretos del modelo predictivo, en donde se hace hincapié en dos cuestiones. La primera relacionada a la representación de la información de entrada de las redes neuronales, la cual se basa en la teoría de word-embeddings, y la segunda relacionada a las teorías a emplear para cada enfoque de clasificación.

3.3.1. Representación de la Información

Cabe destacar que la entrada de las redes neuronales no constituye los mensajes concretos. La representación de los mensajes proviene en forma textual, por lo cual son incompatibles con el formato de entrada de las redes. Existen varios enfoques a utilizar para discretizar la información, para compatibilizarla con la entrada de los modelos predictivos. El clásico enfoque es la bolsa de palabras, que consiste a grandes rasgos en un vector que contabiliza la cantidad de ocurrencias de cada palabra, dado un vocabulario. Dicho método es bastante limitado debido a que no contempla información sobre la relación entre las palabras; sólo se limita a contarlas.

La representación por la cual se optó fue Word-Embeddings, que consiste en representar cada palabra con un vector de longitud fija. Cada elemento de dicho vector, indica la relación de intensidad de la palabra original, con otro concepto. Por ejemplo, si se cuenta con un vector de conceptos para el cual el elemento de la i-ésima posición corresponde al concepto “es hombre”, la palabra “rey”, tendrá un valor de 0.99 en la i-ésima posición de su vector correspondiente. A partir de este enfoque, se pueden relacionar palabras, que en caso de haber utilizado bag of words, no hubiese sido posible. En caso de querer profundizar en el concepto de word-embeddings, en el Capítulo 2 se realiza una explicación detallada, junto con el resto de la base teórica empleada.

Como los mensajes que se quieren clasificar consisten en documentos (oraciones), y no palabras únicas, se optó por tomar la idea de word-embeddings y adaptarla a este tipo de problemática. El procedimiento utilizado fue la elaboración de un promedio de embeddings para cada oración. Como cada oración se puede modelar como una lista de palabras, cada palabra es mapeada a su correspondiente embedding. De esta manera, la lista de palabras original pasa a ser una lista de embeddings. Por último, se toma la lista de embeddings, y se la promedia, para así obtener un solo embedding, que representa a una oración. Así se constituyó la entrada para las redes neuronales, siendo vectores de longitud fija, representando a cada oración, como puede observarse en la Figura 3.8.

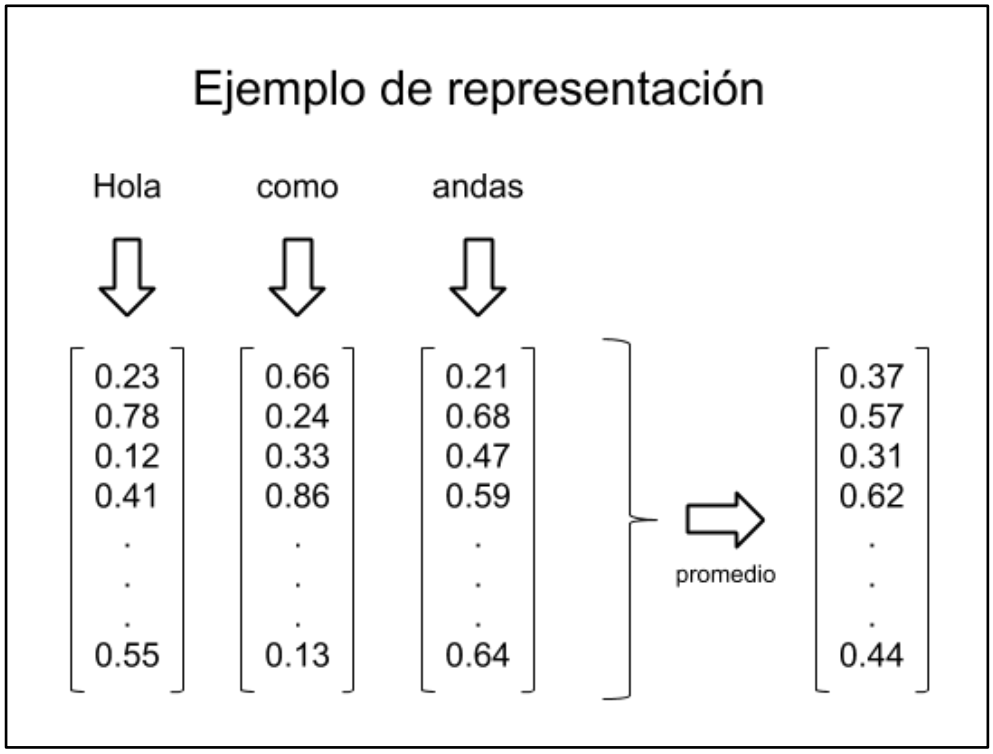

Figura 3.8. Ejemplo ilustrativo de representación de documentos. La idea de word-embedding se extiende a oraciones elaborando un promedio, a partir del embedding asociado a cada palabra.

3.3.2. Técnicas a Emplear

Los enfoques fueron todos desarrollados bajo la misma idea de clasificación por etapas, variando únicamente en la arquitectura empleada para el desarrollo de dichos clasificadores. Los primeros dos enfoques consisten en redes neuronales feedforward. Este tipo de redes están formadas por neuronas que no forman ciclos. El primer enfoque consiste en redes neuronales simples, es decir, redes que aplican funciones de activación sobre solamente capas densas. Además de dichas capas, se emplean otras capas cuyo objetivo es reducir el impacto negativo que afecta a todas las aplicaciones de aprendizaje profundo: el overfitting. Dentro de éste área gran parte del trabajo es combatir a dicho suceso, el cual puede causar graves problemas en caso de probar los clasificadores en producción, ya que se utilizarían ejemplos que no pertenecen a la distribución de datos con la cual se entrenó el clasificador. Las capas utilizadas fueron las conocidas de Batch Normalization y Dropout, que si bien el objetivo de las mismas es reducir el overfitting, generan otras cuestiones que se discuten en detalle en el Capítulo 5, relacionado a la implementación concreta del modelo predictivo.

El segundo enfoque empleado, también consiste en redes neuronales feedforward, aunque en este caso, se trabaja no sólo con capas densas, sino que se recurre a un concepto muy común en el área de Computer Vision: la convolución. Las redes neuronales convolucionales son muy útiles para la extracción de características en los datos que se quieren clasificar. Si bien intuitivamente se puede suponer que la utilización de las mismas se limita solamente a clasificación de imágenes, se ha probado que igualmente son útiles para su utilización en word-embeddings. En el Capítulo 2 se realiza un repaso sobre los trabajos relacionados, en donde entre ellos se hace mención a diversos trabajos de redes convolucionales a partir de word-embeddings.

Luego se experimenta con un enfoque de redes neuronales recurrentes. Este tipo de redes se diferencian de los dos anteriores por poseer ciclos entre sus nodos. Dicha característica es determinante para detectar ciertas dependencias dentro de una misma instancia (en este caso un mensaje), lo cual es de suma importancia en casos de clasificación de texto.

Dentro de las redes recurrentes, se destacan dos tipos de celdas: LSTM y GRU. La segunda es más simple que la primera en cuanto a necesidad de cómputo, por lo tanto, es más escalable. Pero, por otro lado, las LSTM suelen ser mejores debido a mayor capacidad de aprendizaje, constituyendo la opción por defecto con la cual experimentar. Dadas las características del problema a resolver, y al volumen de información con el cual se maneja, se experimentará con LSTM.

Por último, se agrega un cuarto enfoque para la experimentación, empleando los conceptos de los últimos dos enfoques en la misma arquitectura de red neuronal. Esto quiere decir que se desarrollará un modelo predictivo utilizando capas convolucionales y recurrentes. La idea detrás de esta decisión radica en que las convoluciones son muy interesantes para la extracción o detección de características en la entrada de los modelos. Por otro lado, en los patrones que descubran las capas convolucionales, es posible que aún existan dependencias entre las características de un mismo ejemplo. En ese punto entran las capas recurrentes.

Las arquitecturas de redes neuronales generalmente pueden dividirse en dos secciones. Mientras que la primera se encarga del tratamiento y descubrimiento de patrones, la segunda se encarga de la clasificación propiamente dicha. Con respecto a esa clasificación, siempre se utilizan redes neuronales planas, es decir, capas densas. Lo que luego varía, en cada enfoque propuesto, es la primer sección de la arquitectura, que es lo que constituye cada uno. Para el segundo enfoque se utilizó redes convolucionales, y en el tercero recurrentes. Para la segunda sección de la arquitectura de cada enfoque, se utilizan capas densas para realizar la clasificación final para obtener la categoría asociada a cada mensaje. La Figura 3.9 explica de forma gráfica la apariencia que suelen tener las arquitecturas de redes neuronales, cuya forma de plantearse es la adoptada en la solución propuesta.

Cabe destacar que, con respecto al planteo original de utilizar clasificación por etapas, se requieren cinco clasificadores. Los enfoques planteados se utilizan para los cinco clasificadores por motivos de simpleza, evitando complejizar la experimentación al probar todas las combinaciones posibles. Por ejemplo, para el enfoque de convolucionales, se utilizó el mismo enfoque para los cinco clasificadores, procediendo de forma análoga para el resto de los enfoques.

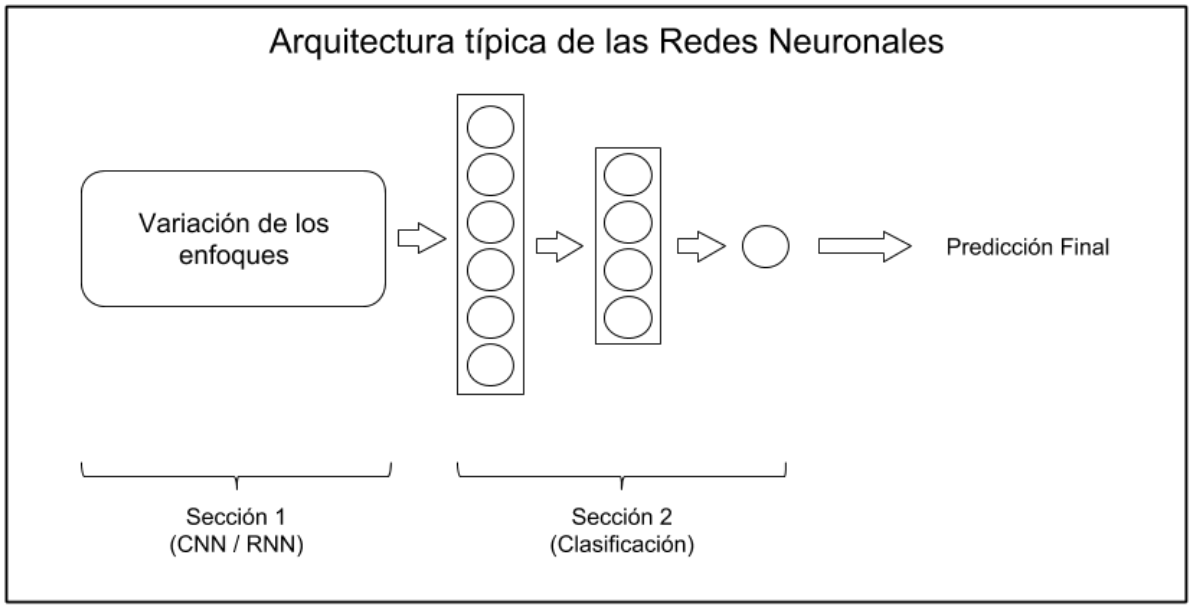

Figura 3.9. A lo largo de los enfoques propuestos, la variación que se introduce corresponde a la Sección 1, en donde se experimenta con redes convolucionales (CNN), recurrentes (RNN) y combinaciones de ellas (CRNN). La segunda sección, encargada de clasificar, debe estar siempre presente al estar trabajando con un problema de clasificación de texto.

3.4. Conclusión

En este capítulo se efectuó el desarrollo de la propuesta de solución para el presente trabajo de grado. Dicha propuesta consistió en una aplicación web que se encarga de generar perfiles de usuario, a partir de un conjunto de mensajes provisto por el usuario de la herramienta. En el

conjunto de mensajes se pueden incluir mensajes de diferentes sesiones de trabajo, lo cual permite observar los perfiles de usuario para cada individuo, para cada sesión en donde participó. De esta manera a través de gráficos y tablas pudo apreciar las diversas fluctuaciones que se puedan dar en los indicadores, a lo largo del tiempo. De esta manera se dejó asentado la visión general de la herramienta para que en el capítulo siguiente se pueda atacar de lleno la implementación de la aplicación.

En la primer sección se cubrieron detalles relacionados a la estructura de la herramienta, separando la explicación de la misma en la aplicación web, y en la aplicación que corre por detrás. Además de la posibilidad de obtener los perfiles de usuario a partir del conjunto de mensajes que provea un usuario, se mostró que la aplicación también ofrece la posibilidad de persistir remotamente los resultados de la clasificación, a fin de no tener que reclasificar todo el dataset en cada momento que el usuario lo necesite. Esto tiene especial relevancia en aquellos casos donde el dataset es muy extenso, lo cual se asocia a tiempos de espera más elevados.

En la Sección 3.2. se descendió un nivel más en la abstracción, en donde además de haber comentado el proceso de desarrollo del modelo predictivo, se proveen detalles de la estructura del modelo. Ya que al tener que predecir conductas, las mismas pueden agruparse en reacciones, lo cual permite el desarrollo de mejores modelos predictivos. En una primer etapa se dispuso de un clasificador de reacciones, y en la segunda etapa se tuvo otro clasificador para las conductas agrupadas bajo la reacción correspondiente. Luego se tuvo que, a partir de las conductas, se las mapea hacia los indicadores Symlog, resultando en los perfiles de usuario finales.

Por último, en la Sección 3.3., se descendió otro nivel más en el desarrollo del modelo predictivo. Se comentó que se realizan experimentaciones empleando diversas teorías de aprendizaje profundo, pasando desde las redes más básicas, hasta modelos híbridos que emplean tanto redes convolucionales como recurrentes dentro del mismo modelo. Además, se efectuó una introducción de la representación de la información a emplear, la cual se basa en la teoría de word-embeddings, en donde en este caso se va a tener a un vector de longitud fija que representa a una determinada oración o mensaje.

Capítulo 4: Implementación de la Herramienta

En este Capítulo se van a discutir los detalles de implementación inherentes a la solución propuesta en el Capítulo 3. Se abarcan diversos aspectos en profundidad, como por ejemplo la arquitectura de la solución empleada, junto con los drivers arquitectónicos que motivaron las decisiones tomadas. Se discutirá en profundidad las tecnologías empleadas, elaborando un análisis del dominio que justifique la motivación de las decisiones tomadas. En el presente capítulo no se hará hincapié en diversas decisiones tomadas en cuanto a la funcionalidad, sino que se realiza especial énfasis en los diversos aspectos relacionados puramente a contenidos vistos durante la carrera de grado, más relacionados con los detalles de la implementación de la solución.

Las Secciones 4.1. y 4.6 hacen referencia a los requerimientos funcionales y escenarios de calidad, respectivamente. En Sección 4.2. se profundiza sobre la arquitectura empleada. Se detalla la funcionalidad e implementación de cada uno de los servicios subyacentes que componen a la arquitectura orientada a servicios. Luego en la Sección 4.3. se describen los endpoints asociados a cada servicio, a fin de comprender correctamente la funcionalidad que cada servicio expone. En la Sección 4.4. se desciende un nivel más en la abstracción, es decir que se abordan las tecnologías empleadas. Por último, en la última sección se detalla la injerencia que cada una de las tecnologías empleadas tiene en la implementación de cada servicio.

4.1. Requerimientos Funcionales

En la presente sección se efectúa un listado correspondiente a la funcionalidad que debe proveer la herramienta desarrollada, mencionando también aquellas cuestiones básicas que motivan la especificación de cada ítem del listado. Los requerimientos funcionales se detallan a continuación:

- La herramienta debe ser capaz de obtener los mensajes a partir de distintos formatos de archivos, a saber: CSV, ARFF y JSON (Takeout).
- La herramienta debe ser capaz de obtener los mensajes a partir de una base de datos (local o remota) basada en MongoDB, con un formato específico.
- La herramienta debe ser capaz de clasificar los mensajes obtenidos a través de diversos enfoques de redes neuronales, para poder constatar la diferencia en la exactitud de cada uno.
- Los resultados que se obtengan a partir de la clasificación, deben ser mostrados de forma intuitiva, a través de tablas y gráficos. Las tablas deben mostrar cada perfil de usuario agrupando todas sus sesiones, mientras que los gráficos deben mostrar cada perfil en cada sesión de trabajo, estando estas sesiones ordenadas por tiempo.
- Los resultados mostrados en la interfaz de usuario, deben poder ser persistidos en una base de datos remota a fin de evitar recaer en reclasificaciones del mismo dataset.
- El sistema debe poder mostrar las clasificaciones realizadas anteriormente, pero de forma actualizada. Es decir que, en caso de haber persistido un conjunto reciente de resultados, la disponibilidad para poder consultar los mismos debe estar garantizada. Esto toma especial

relevancia en caso de que se actualice la página del navegador que contenga la aplicación web.

- La consulta de clasificaciones realizadas anteriormente, debe poder filtrarse por integrante. Es decir que el sistema debe poder consultar los resultados, pero para un determinado integrante. De esta manera se efectúa una recuperación más eficiente de los datos a consultar.

4.2. Arquitectura de la Solución

Si bien, a grandes rasgos, se trata de una arquitectura cliente-servidor, del lado del backend se tomaron ciertas decisiones que facilitaron la extensibilidad del framework a fin de optimizar la compatibilidad del mismo con trabajos futuros. Originalmente se partió de una herramienta de clasificación que emplea una arquitectura monolítica, lo cual contrajo una serie de desafíos a resolver para poder garantizar los atributos de calidad mencionados anteriormente, entre otros. El desarrollo del backend fue materializado a partir de un enfoque orientado a microservicios, a fin de garantizar el desacoplamiento y reusabilidad de cada uno de los componentes o servicios. En las subsecciones que se detallan a continuación, se realiza una descripción de cada uno de los servicios que compone a la arquitectura resultante.

El punto de partida para el desarrollo de la solución propuesta, fue una aplicación desarrollada como parte de la evaluación final de la materia Programación Orientada a Objetos. La misma se encarga de, entre otras cosas, recibir la información y convertir el formato de la misma a uno entendible por las redes neuronales, para luego efectuar la clasificación propiamente dicha. Con la salida de las redes, el mismo framework se encarga de realizar un parsing sobre los mensajes clasificados, para así obtener la conducta necesaria para armar los indicadores correspondientes. Una vez que se lleva a cabo dicho flujo de información, el backend se encarga de devolver un JSON con los indicadores asociados, correspondientes con los perfiles de usuario. Este JSON luego sería parseado por un cliente (front-end) para que así los perfiles de usuario generados se muestren de forma clara y legible para el usuario final. En la Figura 4.1 se muestra un diagrama de la arquitectura que implementa la aplicación web desarrollada. A partir de la exposición de la arquitectura, se procede a analizar dicho diagrama en profundidad, comenzando por detallar la funcionalidad de forma abstracta, para luego ir descendiendo progresivamente en cuanto al nivel de detalle.

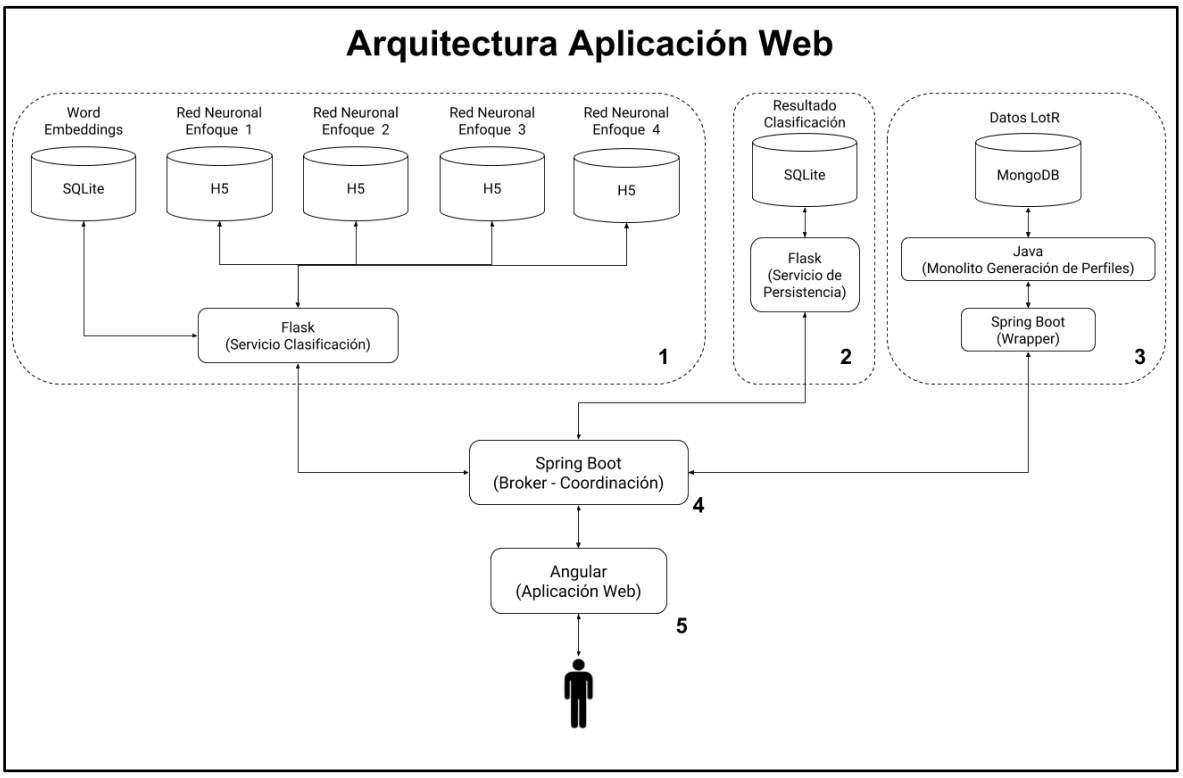

Figura 4.1. Diagrama de arquitectura de la aplicación web que utiliza el framework de modelado de usuarios

Como se puede observar en la Figura 4.1, se tienen cinco servicios o aplicaciones, a saber:

1. Servicio de Clasificación (Redes Neuronales)
2. Servicio de Persistencia (Base de Datos)
3. Servicio Wrapper del Monolito (Conversión de formatos y Generación de Perfiles)
4. Servicio de Coordinación (Broker)
5. Aplicación Web (Front-End)

A continuación, se realiza una descripción detallada del funcionamiento e implementación de cada uno de estos elementos.

4.2.1. Clasificación

El primer servicio está encargado de realizar la clasificación empleando los distintos enfoques de redes neuronales propuestos anteriormente. El funcionamiento consiste básicamente en recibir un CSV, el cual posee, entre otras cosas, un mensaje en cada línea. El resto de los datos que posee no es relevante para este servicio, como por ejemplo la marca de tiempo o el remitente de cada mensaje. A partir del CSV de entrada, se efectúa la predicción, retornando el mismo CSV, pero con la categoría agregada al final de cada línea original. El proceso predictivo está determinado como un flujo secuencial a partir de cada mensaje leído del CSV. Primero se parte de la cadena de texto

correspondiente a cada mensaje de cada línea del CSV. Dicha cadena de texto es traducida a un vector de longitud fija, para así poder someter el mensaje al clasificador. Esta traducción está guiada por el concepto de word-embedding, cuyo funcionamiento fue explicado como parte de la solución propuesta, en el Capítulo 3. Como un word-embedding se corresponde a una sola palabra, lo que se realiza es mapear cada palabra del mensaje a clasificar, con su correspondiente embedding, para después promediar los vectores, obteniendo como resultado un "sentence-embedding" que represente a una oración, es decir al mensaje original.

Luego, una vez que se obtiene la representación compatible con el clasificador, se somete el embedding al clasificador de reacciones, que se corresponde con la primer etapa de la clasificación. La reacción predicha en este primer paso, determina por cuál de los cuatro clasificadores volverá a ser sometido el embedding, ya que cada clasificador de la segunda etapa predice una de tres conductas posibles, correspondientes a la reacción predicha. A modo de ejemplo, se tiene a un determinado embedding que, en su primer etapa, fue clasificado con la reacción positiva. Dicha predicción provoca que el mismo embedding sea sometido (ahora en la segunda etapa), al clasificador encargado de predecir las conductas positivas. Es decir, que el clasificador que le tocará al embedding en segunda instancia, hará una predicción sobre tres conductas posibles que son: "Muestra solidaridad (C1)", "Muestra relajamiento (C2)", y "Muestra acuerdo (C3)". Dicho ejemplo se puede ver plasmado en la Figura 4.2 en donde se incluye el proceso que realiza el servicio para efectuar la clasificación.

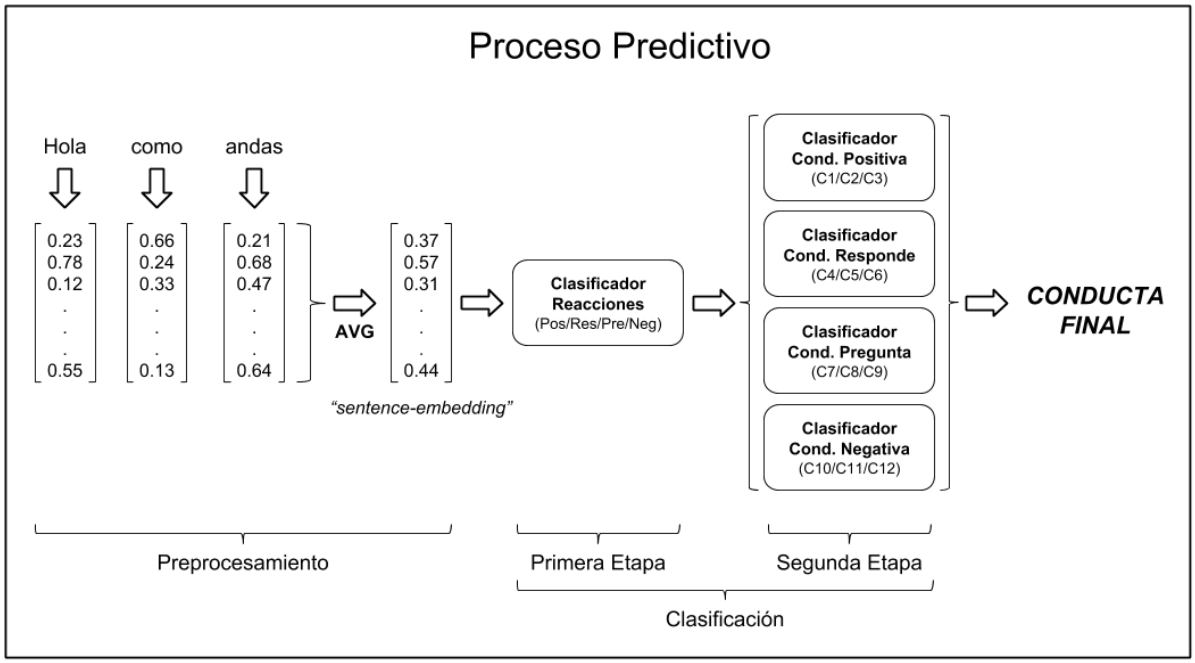

Figura 4.2. Pasos llevados a cabo por el servicio, tomando como punto de partida un determinado mensaje. Este proceso se repite tantas veces como mensajes tenga el CSV de entrada. Los embeddings mostrados son ilustrativos, no se corresponde con su representación vectorial real.

Recientemente se mencionó que cada línea del CSV posee el mensaje, entre otros datos. Esta información adicional, irrelevante para el servicio actual, consiste en el remitente del mensaje, identificador de la sesión de chat en donde se envió dicho mensaje, y la marca de tiempo de la sesión en donde se encuentra la sesión. De todas maneras, estos datos se incluyen debido a que posteriormente se utiliza por el framework para generar los perfiles asociados.

Con respecto a los detalles de las redes neuronales, en esta sección se hace referencia a ellas como si de una caja negra se tratase. En el Capítulo 5 se detalla la arquitectura de cada una en profundidad, abarcando desde el tipo de capas utilizadas, hasta los parámetros empleados para llevar a cabo los entrenamientos de las mismas.

4.2.2. Persistencia

El segundo servicio es el encargado de almacenar los resultados de las clasificaciones realizadas. Esto se debe a la necesidad de que el usuario no debiera reclasificar cada vez que quiera consultar los perfiles nuevamente, ya que penalizaría gravemente la usabilidad de la aplicación, sobre todo en contextos donde la cantidad de mensajes a clasificar es muy grande. Se provee la posibilidad de que, una vez que un conjunto de mensajes fue clasificado, los perfiles generados puedan ser persistidos en una base de datos remota.

Este microservicio posee una complejidad de implementación mucho menor que el de clasificación. Solo se encarga de almacenar los resultados en una tabla cuyo nombre es ingresado por el usuario final, y de consultar dichos datos a través del nombre de la tabla. Una función tardía que le fue añadida, por cuestiones de optimización de usabilidad de la aplicación web, es la posibilidad de retornar los nombres de los resultados persistidos. Esto otorga la posibilidad de que dichos nombres puedan ser incluidos en la interfaz de usuario del lado del cliente, para que así se pueda saber en todo momento las tablas que hay almacenadas.

Las tecnologías empleadas para materializar dicho microservicio se exponen en la Sección 4.4, al igual que para el resto de los servicios. En esta sección se explica el funcionamiento de cada uno, pero sin recaer en el apartado tecnológico a fin de separar la funcionalidad de los detalles del workflow de cada tecnología.

4.2.3. Gestión de Formatos y Generación de Perfiles

El tercer servicio resulta ser el más grande del listado. Consiste en una herramienta de clasificación mediante técnicas de aprendizaje automático tradicional, desarrollada para la materia Programación Orientada a Objetos, correspondiente al tercer año de la carrera. Dicha herramienta fue desarrollada sin tener conocimiento básico sobre patrones arquitectónicos, atributos de calidad, etc. El resultado fue una herramienta con un diseño que naturalmente resultó ser monolítico, con los riesgos que ello conlleva a la hora de escalar y extender la funcionalidad. Entre dichos problemas, se puede observar la acumulación de funcionalidad, idea contraria al enfoque arquitectónico que se está llevando a cabo para el desarrollo del sistema. Otra complicación que puede surgir a partir del uso de monolitos es la complejidad del funcionamiento de los mismos. Mientras que una

arquitectura orientada a microservicios tiende a separar la funcionalidad para mejorar ciertos aspectos como por ejemplo la escalabilidad, modificabilidad e interoperabilidad, el monolito desde su concepción complica el tratamiento de estos atributos de calidad. Dicha complicación se hace más notoria al querer diseñar una aplicación extensible, con vistas a su aprovechamiento para trabajos futuros.

Volviendo a la funcionalidad de la herramienta, el trabajo realizado con la misma consistió en "envolverla" con un framework especializado para el desarrollo de servicios. De esta forma se expuso solo la funcionalidad necesaria a través del aprovechamiento de parte de la implementación realizada anteriormente. La funcionalidad original de dicho trabajo fue el empleo de un clasificador de texto realizado por Martín Mineo en su trabajo final de grado. Dicho clasificador emplea técnicas tradicionales de clasificación de texto basadas en Support Vector Machines (SVM). En el caso del presente trabajo de grado, dicho clasificador es reemplazado por los enfoques de aprendizaje profundo, bajo la hipótesis de que se obtendrán mejores resultados. Los detalles inherentes a la experimentación y a los resultados obtenidos se abordan en el Capítulo 6.

Por otro lado, este servicio es el encargado de alojar la funcionalidad que materializa una parte del framework descrito anteriormente: la generación de perfiles a partir de los mensajes clasificados. A partir de un archivo CSV que se le ingresa como entrada (que fue el que retornó la red neuronal, con los mensajes ya clasificados), el servicio se encarga de parsear dicho archivo para generar los perfiles. Anteriormente, se estableció que cierto tipo de información del CSV era irrelevante para el servicio de la red neuronal (marca de tiempo, remitente, etc.). Dicha información se utiliza en este servicio. El identificador de sesión de chat se utiliza para agrupar a los integrantes de cada conversación. La marca de tiempo para poder ordenar las sesiones de chat. Por último, de forma trivial se puede decir que el remitente de cada mensaje es necesario para que finalmente se le puedan asociar la suma de sus clasificaciones de sus conductas y sus indicadores Symlog.

Una vez que el servicio generó los perfiles asociados a cada usuario presente en el conjunto de mensajes provisto, retorna un archivo con formato JSON con dicha información. Se utiliza este formato por dos motivos. El primero radica en que no es un formato "final", es decir, que el usuario final no ve esta salida, sino que un cliente debe efectuar un parsing sobre dicho JSON para poder hacer una presentación amena de los perfiles generados. JSON es un formato que tiene una relación natural con el lenguaje Typescript, muy presente del lado del cliente, por lo que dicho formato es muy sencillo de parsear utilizando dicho lenguaje, que resulta ser el escogido para la implementación del cliente. Por otro lado, la tecnología empleada para el desarrollo de algunos servicios, Spring Boot, tiene como característica facilitar el armado de los JSON, a través de mapeos concretos con las clases Java que representan los perfiles generados. De esta manera, la tarea de trabajar con el JSON resulta ser sumamente sencilla, tanto para producirlo como para consumirlo. La elección de la tecnología es fundamental para este tipo de características. Habiendo introducido algunas de las tecnologías utilizadas, los detalles de cómo se empleó cada una se reservan para la Sección 4.5, dedicada especialmente para este tipo de cuestiones relacionadas a tecnología. Para clarificar el trabajo realizado por el servicio, se muestra en la Figura 4.3 un ejemplo de generación de perfiles, en donde el usuario final ingresó mensajes en formato ARFF.

Como se puede observar, el identificador de la conversación es el nombre del archivo ARFF que el usuario final envió. Cabe remarcar que este JSON no es el que finalmente observa dicho usuario, sino que el cliente debe parsear este archivo para mostrárselo de forma amigable al usuario. Una observación a realizar, es que, en este caso, por cuestiones de simplicidad en la ejemplificación, se adjuntó un solo archivo ARFF. Prueba de esto es que en la captura se puede ver que el primer caracter de la salida es un corchete: se trata de un arreglo. Dicho arreglo contendrá tantos elementos como conversaciones haya para clasificar.

ejemplo_clasificacion.json


```
[
  {
    "idChat": "UgwNrPND3WHLDnr1s_t4AaABAQ.arff",
    "mapeoPersonaClasificacion": {
      "Integrante_1": {
        "roles": [0.5, 0.0, 3.5, 2.0, 0.0, 0.5, 1.5, 1.0, 2.0],
        "conductas": [3, 0, 0, 1, 2, 2, 2, 0, 0, 0, 0],
        "reacciones": [3, 5, 2, 0],
        "conductasPorcentual": [30.0, 0.0, 0.0, 10.0, 20.0, 20.0, 20.0, 0.0, 0.0, 0.0, 0.0, 0.0],
        "indicadoresSymlog": [2, 8, 3, 0, 7, 3],
        "reaccionesPorcentual": [30.0, 50.0, 20.0, 0.0],
        "rolesPorcentual": [4.55, 0.0, 31.82, 18.18, 0.0, 4.55, 13.64, 9.09, 18.18]
      },
      "Integrante_2": {
        "roles": [0.5, 0.0, 7.0, 4.5, 0.0, 0.5, 3.5, 2.0, 6.0],
        "conductas": [4, 0, 3, 1, 6, 3, 4, 1, 0, 0, 0],
        "reacciones": [7, 10, 5, 0],
        "conductasPorcentual": [18.18, 0.0, 13.64, 4.55, 27.27, 13.64, 18.18, 4.55, 0.0, 0.0, 0.0, 0.0],
        "indicadoresSymlog": [5, 17, 7, 0, 15, 7],
        "reaccionesPorcentual": [31.82, 45.45, 22.73, 0.0],
        "rolesPorcentual": [2.08, 0.0, 29.17, 18.75, 0.0, 2.08, 14.58, 8.33, 25.0]
      },
      "Integrante_3": {
        "roles": [0.0, 0.0, 7.5, 6.0, 0.0, 0.0, 2.5, 1.0, 6.5],
        "conductas": [2, 0, 1, 0, 5, 7, 2, 1, 0, 0, 0],
        "reacciones": [3, 12, 3, 0],
        "conductasPorcentual": [11.11, 0.0, 5.56, 0.0, 27.78, 38.89, 11.11, 5.56, 0.0, 0.0, 0.0, 0.0],
        "indicadoresSymlog": [3, 15, 3, 0, 15, 3],
        "reaccionesPorcentual": [16.67, 66.67, 16.67, 0.0],
        "rolesPorcentual": [0.0, 0.0, 31.91, 25.53, 0.0, 0.0, 10.64, 4.26, 27.66]
      },
      "Integrante_4": {
        "roles": [3.0, 0.0, 11.0, 4.5, 0.5, 3.0, 4.0, 1.5, 9.5],
        "conductas": [9, 0, 10, 6, 2, 7, 3, 4, 0, 1, 0, 0],
        "reacciones": [19, 15, 7, 1],
        "conductasPorcentual": [21.43, 0.0, 23.81, 14.29, 4.76, 16.67, 7.14, 9.52, 0.0, 2.38, 0.0, 0.0],
        "indicadoresSymlog": [8, 34, 19, 1, 22, 20],
        "reaccionesPorcentual": [45.24, 35.71, 16.67, 2.38],
        "rolesPorcentual": [8.11, 0.0, 29.73, 12.16, 1.35, 8.11, 10.81, 4.05, 25.68]
      },
      "Integrante_5": {
        "roles": [0.0, 0.0, 3.0, 1.0, 0.0, 0.0, 0.5, 0.5, 1.5],
        "conductas": [4, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0],
        "reacciones": [5, 2, 1, 0],
        "conductasPorcentual": [50.0, 0.0, 12.5, 0.0, 12.5, 12.5, 12.5, 0.0, 0.0, 0.0, 0.0, 0.0],
        "indicadoresSymlog": [1, 7, 5, 0, 3, 5],
        "reaccionesPorcentual": [62.5, 25.0, 12.5, 0.0],
        "rolesPorcentual": [0.0, 0.0, 46.15, 15.38, 0.0, 0.0, 7.69, 7.69, 23.08]
      }
    },
    "timestamp": "2015-12-16T20:20:00",
    "integrantes": [
      "Integrante_1",
      "Integrante_2",
      "Integrante_3",
      "Integrante_4",
      "Integrante_5"
    ],
    "rolesGrupales": [4.0, 0.0, 32.0, 18.0, 0.5, 4.0, 12.0, 6.0, 25.5],
    "conductasGrupales": [22, 0, 15, 8, 16, 20, 12, 6, 0, 1, 0, 0],
    "reaccionesGrupales": [37, 44, 18, 1],
    "conductasGrupalesPorcentual": [22.0, 0.0, 15.0, 8.0, 16.0, 20.0, 12.0, 6.0, 0.0, 1.0, 0.0, 0.0],
    "reaccionesGrupalesPorcentual": [37.0, 44.0, 18.0, 1.0],
    "rolesGrupalesPorcentual": [3.92, 0.0, 31.37, 17.65, 0.49, 3.92, 11.76, 5.88, 25.0]
  }
]
```


Figura 4.3. Ejemplo de generación de perfiles para una conversación en formato ARFF.

A continuación del identificador de chat, se encuentran las clasificaciones individuales, cuyos nombres fueron anonimizados. Como se puede observar, se muestran las conductas predichas por las redes neuronales, las reacciones (capa de abstracción sobre las conductas), los indicadores Symlog, y además se incluye los mismos indicadores, pero de forma porcentual, a fin de facilitar otro tipo de métricas que se quieran realizar sobre esta salida.

Luego de los perfiles individuales, se muestra la marca de tiempo que caracteriza a la conversación clasificada, con respecto a este dato no hay más información por aclarar. Se incluye también un listado de los integrantes que componen a la sesión de chat clasificada. Por último, se incluyen los mismos indicadores que para cada individuo, pero de forma grupal, es decir para cada conversación. Esto permite obtener más indicadores de cómo ha resultado la interacción en la sesión clasificada. En el ejemplo mostrado, se puede apreciar que, para la última reacción, la negativa, se obtuvieron indicadores muy bajos, más precisamente del 1%, lo cual indica que la probabilidad de que se hayan producido situaciones de conflicto en el grupo de trabajo, es muy baja.

De modo aclaratorio, se informa que la clasificación realizada se llevó a cabo enviando un solo archivo ARFF con la clasificación, y con un parámetro extra que indica la cantidad de mensajes a clasificar, que en este caso fue de 100 mensajes. Esto explica que, en caso de sumarse todos los mensajes clasificados en el JSON de respuesta puesto a modo de ejemplo, se obtenga dicha cantidad, lo cual favorece también al cálculo de los valores porcentuales grupales. Los parámetros de cada interacción con los servicios se tratan en la Sección 4.3, relacionada a los Endpoints, es decir, a los puntos de comunicación entre los servicios.

La última funcionalidad expuesta en este servicio, a partir del monolito previamente implementado, es la gestión de los formatos. Debido a funcionalidad existente, se apalanca la representación interna de los conjuntos de mensajes para obtener los CSV correspondientes a cada fuente de información. Este CSV constituye luego la entrada para que luego las redes neuronales se encarguen de la clasificación. De esta manera, se da soporte a otras fuentes de información como por ejemplo el ya mencionado ARFF, también JSON (correspondiente al Google Takeout), y a mensajes provenientes de una conexión con una base de datos que utilice MongoDB

En resumen, el monolito expuesto en forma de servicio realiza dos tareas. La primera es la conversión de formato para los conjuntos de mensajes, para generar el CSV que reciba la red neuronal. La segunda tarea es recibir un CSV clasificado por la red neuronal, para retornar un JSON con todos los perfiles generados a través de las técnicas descritas anteriormente.

4.2.4. Broker

El cuarto servicio desarrollado consiste en un Broker. Dentro de una arquitectura orientada a servicios totalmente desacoplados, lo que se tiene básicamente son distintos componentes que realizan una determinada funcionalidad, pero que por sí mismos no hacen al sistema total, es decir a la aplicación web. Por otro lado, el hecho de contar con todos los servicios completamente aislados y desacoplados, hace que se haga más compleja la integración de la funcionalidad para satisfacer las necesidades del usuario final. Si la idea es implementar un cliente liviano, lo ideal no sería que toda la lógica de procesado de las sucesivas solicitudes se lleve a cabo en el lado del cliente. Este

tipo de situaciones motivan la existencia de un componente central que esté presente entre el cliente y los servicios descritos anteriormente, que implementan la funcionalidad. Este componente se encarga de procesar las solicitudes del cliente y, para proveerle una respuesta, realizar las correspondientes solicitudes sucesivas a cada servicio/microservicio para ir formando la respuesta a retornar. Aplicando el patrón arquitectónico al problema a resolver, la respuesta final que irá "armando" el broker son los perfiles asociados al conjunto de mensajes que el usuario final proporcionó. En la Figura 4.4 se provee un ejemplo de flujo de ejecución completo, en donde inicialmente se proporciona un archivo CSV con los mensajes a clasificar. Cabe aclarar que, por motivos de simplicidad en el ejemplo, el hecho de emplear un CSV para la clasificación implica que no se deban realizar pasos adicionales referidos a conversión de formatos a fin de compatibilizar los mensajes con las redes neuronales.

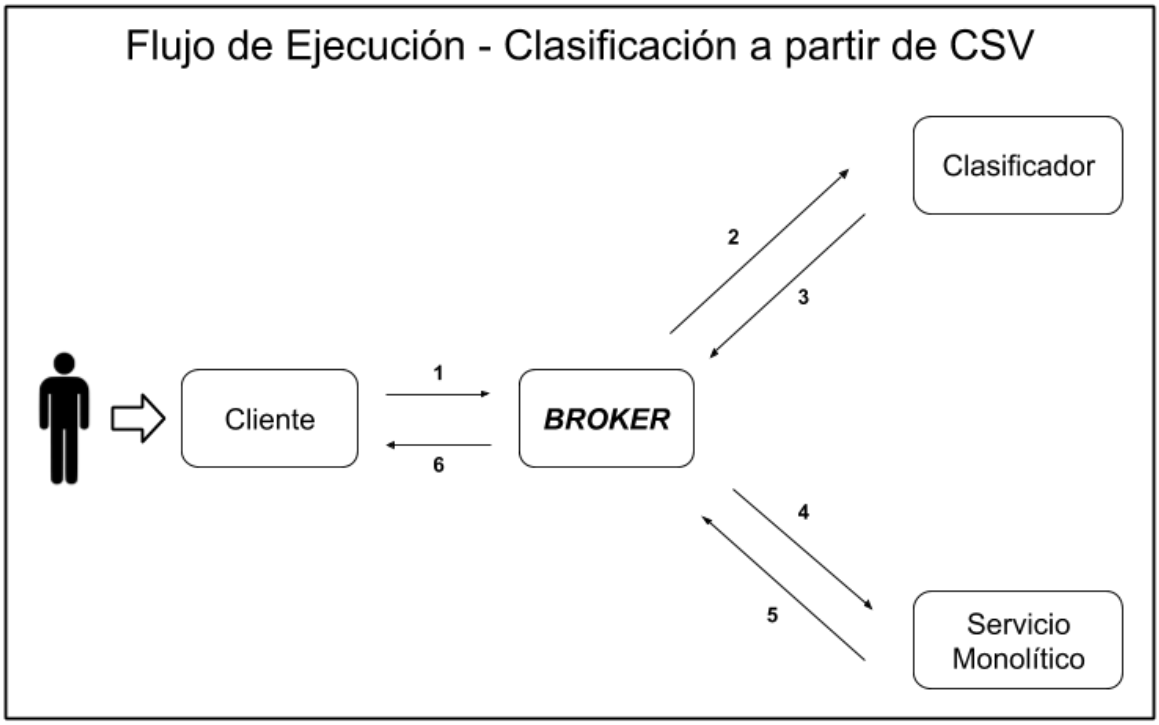

Figura 4.4. Flujo de ejecución básico empleando un CSV como fuente de clasificación, utilizando para ello el patrón arquitectónico Broker.

En el ejemplo provisto de la Figura 4.4 se expone un ejemplo básico cuya numeración en las acciones se describirá detalladamente a continuación:

1. El cliente provee un archivo CSV cuyo contenido consiste en el conjunto de mensajes del cual generar los perfiles asociados.
2. El broker recibe el CSV, y lo reenvía al servicio de clasificación, para obtener las conductas asociadas a cada mensaje.

3. El clasificador retorna al broker el mismo CSV, pero clasificado. Es decir, que a cada línea (que contiene cada mensaje) le agrega un campo más, que se corresponde con la conducta predicha.
4. El broker toma el CSV clasificado, y se lo envía al servicio encargado de armar los indicadores para cada usuario y para cada conversación.
5. El monolito encargado de armar los perfiles de usuario, retorna al broker dicha información de forma estructurada, en formato JSON.
6. El broker toma el JSON y se lo reenvía al cliente, es decir al front-end.

Una vez observado el funcionamiento básico del backend, se puede proceder a casos de uso más complejos, como por ejemplo aquél en donde el usuario final provee una fuente de información que no es la nativa de los clasificadores, lo cual hace que sea necesaria una conversión de formato. Adicionalmente, a fin de proporcionar un ejemplo opuesto al anterior en cuanto a complejidad, se agrega la situación de que el usuario quiera almacenar los resultados de la clasificación en la base de datos remota. Este caso de uso se expone en la Figura 4.5. Esto último provee la posibilidad de que luego el usuario pueda recuperar instantáneamente dichos resultados mediante una consulta a la base de datos, sin tener que recurrir a tener que clasificar el dataset entero. En cuanto a tiempos, en este caso el acceso a base de datos se realiza casi instantáneamente, en comparación a la misma obtención de los resultados, pero a partir de la clasificación de los mismos.

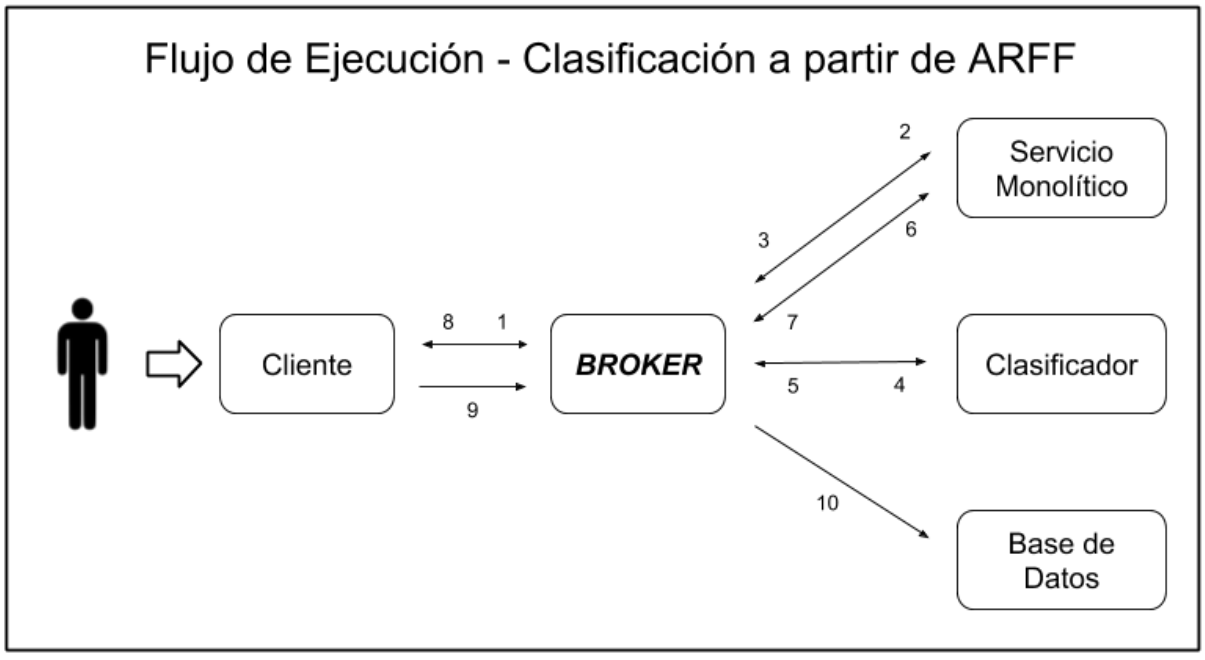

Figura 4.5. Flujo de ejecución básico empleando un ARFF como fuente de clasificación, utilizando para ello el patrón arquitectónico Broker.

En el ejemplo provisto de la Figura 4.5 se expone un ejemplo complejo cuya numeración en las acciones se describirá detalladamente a continuación:

1. El cliente provee al broker un archivo ARFF, junto con un CSV que asocie al identificador de cada conversación con una marca de tiempo. Esto se debe a que el formato ARFF naturalmente no provee este tipo de información, por lo que el usuario final debe proveer este tipo de información para que luego se pueda monitorear la evolución de los indicadores a lo largo del tiempo.
2. Como el formato de los mensajes no es el CSV que necesita el clasificador, se debe realizar un paso extra antes de efectuar la predicción de las conductas. El broker envía el ARFF y el CSV con el mapeo al servicio monolítico.
3. El servicio monolítico retorna un CSV con los mensajes y el resto de la información lista para que sea clasificada. Esta información es enviada al broker.
4. El broker toma el CSV recibido por el servicio monolítico, y lo reenvía al servicio de clasificación, para obtener las conductas asociadas a cada mensaje.
5. El clasificador le retorna al broker el mismo CSV, pero clasificado. Es decir, que a cada línea (que contiene cada mensaje) le agrega un campo más conteniendo la conducta predicha.
6. El broker toma el CSV clasificado, y se lo envía al servicio encargado de armar los indicadores para cada usuario y para cada conversación.
7. El monolito encargado de armar los perfiles de usuario, retorna al broker dicha información de forma estructurada, en formato JSON.
8. El broker toma dicho JSON y se lo reenvía al cliente, es decir al front-end.
9. El front-end envía al broker una solicitud para persistir los resultados obtenidos, es decir la clasificación que recibió anteriormente. El formato en el cual se envía la clasificación es un CSV, en donde cada usuario se corresponde con una fila, y cada columna se corresponde con uno de los tantos indicadores que componen el perfil de usuario.
10. El broker envía una solicitud al servicio de la base de datos, con el CSV que recibió del cliente. Dicho archivo se guarda en una base relacional con el formato sin tocar, es decir que el encabezado del CSV constituyen las columnas de la tabla a dar de alta en la base. Luego la cantidad de filas es la misma que la del CSV.

Como se puede observar en ambos ejemplos, la comunicación del cliente con el backend se ve simplificada al tener que comunicarse el cliente con solamente el broker. Es decir, que toda la comunicación entre los servicios se da a través del broker. De esta manera el cliente solo debe conocer al broker para poder solicitar la funcionalidad. Luego este componente central se encarga de hacer todo el pasaje de información entre los distintos servicios con los que se cuentan para poder efectuar la respuesta al cliente. Cabe destacar que el trabajo del broker no es implementar comportamiento sino orquestrar los diferentes servicios que integren la funcionalidad a ofrecerle al usuario final. A modo introductorio de la tecnología a utilizar para la comunicación entre servicios, se anticipa que para toda comunicación se emplea el protocolo HTTP, a fin de garantizar el total desacoplamiento de cada componente dentro del backend.

Una de las contrapartidas que el enfoque basado en broker puede traer, es que dicho componente puede constituir un único punto de falla, en donde cualquier eventualidad que surja y afecte a la disponibilidad del broker, afectará a toda la funcionalidad. De esta manera el cliente, en caso de caerse el broker, pierde toda posibilidad de acceder al backend, ya que el mismo está totalmente abstraído por el broker. Esta desventaja puede tener más relevancia en casos de sistemas críticos, o en ambientes de producción masivos en donde se debe prestar especial atención a la disponibilidad. En dichos casos, la caída del nodo donde se encuentra alojado el broker puede conllevar importantes gastos económicos para la organización, por lo que se recurre a otro tipo de técnicas de balance de cargas, utilizando mecanismos de replicación y escalabilidad tanto horizontal como vertical a fin de garantizar elevadas métricas de disponibilidad.

Con respecto a la forma concreta de la comunicación entre los servicios, lo que se realiza es una serie de solicitudes HTTP a los endpoints de cada servicio. Dichos endpoints se detallan en su totalidad en la Sección 4.3.

4.2.5. Aplicación Web

Hasta el momento se describieron cuatro servicios, todos relacionados con el backend. Dichos servicios son gestionados por un broker que, a vistas del cliente, se encarga de abstraer toda la funcionalidad del backend. El cliente que se desarrolló se trata de un servicio más, es decir que, ante una determinada solicitud, el servicio envía una página web, que es la que termina siendo vista por el usuario final. Dicho cliente consiste en una interfaz de usuario, que implementa una lógica mínima, encargada de procesar las respuestas que envía el broker. Dicha interfaz está presentada de forma tal que el usuario final no deba perder tiempo entendiendo el formato JSON, sino que la presentación se da a través de tablas y gráficos cómodos para la vista. Una de las premisas principales a la hora del desarrollo de este servicio es que el usuario final, es decir el hipotético supervisor, no deba provenir necesariamente del área de sistemas para poder utilizar fácilmente la aplicación web.

En comparación con el broker, lo que el cliente hace de cierta manera es lo mismo. Ambos exponen la misma funcionalidad, aunque de diferente forma. Mientras el broker lo hace de forma programática, el cliente lo hace a través de la interfaz de usuario. En este caso el usuario lógicamente interactúa con el cliente en vez de con el broker. La funcionalidad ofrecida en este caso comienza con la carga de la información, ejemplificada en la Figura 4.6. Como se describió anteriormente, el usuario debe proporcionar una fuente de información sobre la cual generar los perfiles. Esta fuente de información puede ser proporcionada en varios formatos: ARFF, JSON (Takeout), CSV y a partir de una base de datos que emplee MongoDB. Para este último tipo de fuente, al igual que con el resto de los archivos que deben respetar un formato, la base de datos también debe respetar una estructura específica, para que así posibilite al backend extraer los mensajes de forma correcta.

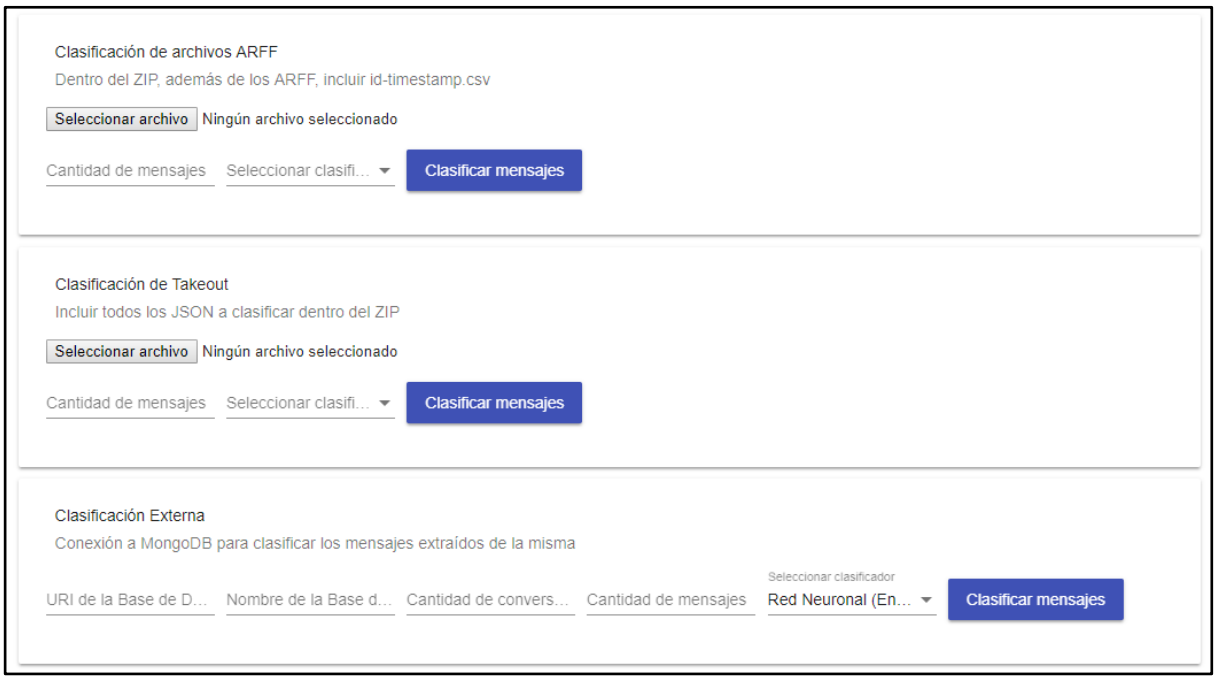

Figura 4.6. Ejemplo de la carga de información a través de los distintos tipos de fuentes de información.

Otro detalle a destacar al momento de la carga de la información, es la posibilidad que se le ofrece al cliente para elegir el enfoque con el cual se clasificarán los mensajes. Entre los clasificadores ofrecidos, se ponen a disposición cada uno de los enfoques de aprendizaje profundo empleados, a fin de monitorear variaciones en el comportamiento de la clasificación de cada enfoque.

Una vez que se envía la fuente de información, en cualquiera de los formatos soportados, el backend responderá con los perfiles asociados a cada usuario y a cada conversación. Dichos perfiles se podrán observar en el cliente de dos maneras. La primera es a través de tablas. Se ofrecen cuatro tablas para poder separar la información a fin de optimizar la legibilidad de los indicadores. El orden de la información que muestran las tablas es el siguiente:

- Conductas IPA (12 columnas)
- Reacciones (Abstracción sobre IPA, 4 columnas)
- Indicadores Symlog (6 columnas)

La segunda forma en la que se manifiestan los perfiles de usuario, es a través de gráficos de líneas por individuo. A partir de un individuo, se selecciona la tabla sobre la cual mostrar la información, es decir que, para un determinado integrante, se pueden elegir mostrar uno de los cuatro tipos de información recientemente listados.

Otras de las funcionalidades a exponer en el cliente, es la posibilidad de persistir los resultados de la clasificación. Antes de enviarle la clasificación al broker, se realiza una lógica mínima para recolectar la información que se muestra en las cuatro tablas. Dicha información es estructurada dentro de un CSV, el cual resulta ser el archivo que se envía a la base para que se persista. La

información se dispone en el CSV de forma tal que se garantice el orden que se mostró de la misma en las tablas. De esta manera, al querer recuperarse posteriormente dicha clasificación, el broker enviará un CSV con la tabla entera de la base de datos, para luego poblar las tablas del cliente.

4.3. Descripción de Endpoints

En esta sección se detalla en profundidad los puntos de exposición de cada servicio. Es decir, los puntos a través de los cuales los servicios van a ser invocados, sea a través del broker o del cliente. El protocolo empleado para la comunicación en todo momento fue HTTP, para todos los servicios, por lo que se realiza un barrido por el funcionamiento y detalle de los controladores de cada uno de ellos.

A continuación, se muestra una tabla con la descripción de cada endpoint, para cada servicio. Asociada a cada tabla se desarrolla una explicación para profundizar y clarificar todos los aspectos de la misma.

4.3.1. Clasificación

Operación	Método HTTP	Parámetros	Salida
Clasificación de CSV	POST	Cantidad de Mensajes	CSV clasificado
		CSV a clasificar	
		Red Neuronal a Emplear	

Tabla 4.1. Exposición HTTP del microservicio encargado de efectuar la clasificación de los mensajes manifestados en formato CSV.

Como se puede observar en la Tabla 4.1, el microservicio de clasificación es relativamente simple en cuanto a exposición del mismo. La única funcionalidad que realiza es la clasificación de los mensajes, a partir de un CSV. Dicha clasificación puede variar mediante dos parámetros que se incluyen en la solicitud, que es la cantidad de los mensajes a clasificar para la fuente provista, y la red neuronal a emplear. En caso de que no se quiera probar la clasificación para todos los mensajes por cuestiones de tiempos de espera, se puede ingresar un valor entero que indique la cantidad de mensajes. Si se quiere clasificar todos los mensajes, se ingresa un “-1” en dicho parámetro a modo de indicador.

Una aclaración a realizar es el método HTTP empleado. La acción que se realiza es, dicho de otra manera, la obtención de las conductas asociadas a cada mensaje, por lo que se trataría de una consulta. La teoría de las arquitecturas REST indica una determinada acción a realizar por cada método [39]. Por ejemplo, un método GET se debe utilizar para consultas de información. Un método PUT indica una actualización de la información. Un método POST indica que se da de alta cierta información. Esto indica en principio que la decisión de emplear un POST para efectuar la clasificación, no fuese del todo acertada. Se encontró a continuación una limitación en las bases teóricas de las arquitecturas REST, al menos en cuanto a la herramienta utilizada como cliente HTTP.

Durante el desarrollo del servicio, se utilizó una herramienta llamada Postman, que se encarga de facilitar el testing de servicios HTTP. Como la tarea a realizar consistió en la clasificación de un archivo CSV, se quiso testear enviando solicitudes GET al servicio. La realidad es que dicha herramienta no lo permitía, argumentando que las bases teóricas de REST sostienen que una solicitud GET no puede tener cuerpo. Esta cuestión implica la imposibilidad de efectuar este tipo de solicitudes a partir de un CSV. La solución tomada fue un workaround en donde se modeló al endpoint de clasificación para que reciba solicitudes POST, aunque no fuese teóricamente correcto. De esta manera, se pudo finalmente enviar archivos CSV en el cuerpo de estas solicitudes

4.3.2. Persistencia

Operación	Método HTTP	Parámetros	Salida
Obtención de Resultados	GET	Nombre de Tabla	CSV con los resultados de la tabla solicitada
Obtención de Resultados	GET	Nombre de Tabla Nombre Integrante	CSV con los resultados de la tabla solicitada para el integrante solicitado
Obtención de Resultados Disponibles	GET	(sin parámetros)	Listado con el nombre de los resultados
Almacenar Resultados	POST	CSV con los resultados de la clasificación	Confirmación del alta realizada

Tabla 4.2. Exposición HTTP del microservicio encargado de efectuar y consultar la persistencia de los mensajes previamente clasificados.

Como se comenta en el epígrafe de la Tabla 4.2, los métodos que se exponen para trabajar con el servicio están inherentemente relacionados con el alta y consulta de la información, es decir de los

resultados de la clasificación. La consulta puede realizarse por tabla entera, o bien filtrando por integrante, con los ahorros que eso conlleva en términos de performance. Otra consulta que se incluye, para evitar que el usuario final deba recordar los nombres de las tablas, es un GET que retorna un listado de los nombres de las tablas almacenadas en la base de datos. Por último, se dispone de un método POST en donde se guarda el CSV enviado en el cuerpo de la solicitud, que se corresponde con los resultados mostrados en las tablas del lado del cliente.

4.3.3. Gestión de Formatos y Generación de Perfiles

Operación	Método HTTP	Parámetros	Salida
Obtener CSV a partir de ARFF	POST	Archivo ZIP con los ARFF y el CSV de los timestamp	CSV con el dataset formateado
Obtener CSV a partir de Takeout	POST	Archivo ZIP con los JSON	CSV con el dataset formateado
Obtener CSV a partir de BD	GET	URI de la BD Nombre de la BD Cantidad chats	CSV con el dataset formateado
Generar los perfiles	POST	Archivo CSV clasificado	JSON con los perfiles generados

Tabla 4.3. Exposición HTTP del servicio encargado de efectuar las conversiones a formato CSV, y al armado de los perfiles de usuario a partir de los CSV clasificados.

Los primeros tres endpoint se corresponden con la gestión de formatos que realiza el servicio. Como las redes neuronales trabajan con cierta estructura de tipo CSV, este servicio se encarga aplicar un denominador común a todas estas fuentes. Se aprovecha la funcionalidad de este monolito con respecto el manejo de datasets, para generar un CSV listo para ser clasificado. La llamada a estos endpoints es realizada por el broker, que únicamente se encarga de ser un intermediario al momento de enviar y recibir información, por lo que el broker, al recibir el CSV que retorna este servicio, lo reenvía directamente al servicio del clasificador. En este caso también aplica la cuestión surgida durante el desarrollo del primer servicio, el de clasificación. Como en teoría no se puede incluir un cuerpo para una petición GET, en aquellas solicitudes en donde se necesitaba adjuntar un archivo a la solicitud, se empleó un POST.

La segunda parte de la funcionalidad del servicio es la generación de perfiles. A partir de un CSV clasificado, es decir a partir del retorno del clasificador de redes neuronales, se apalanca la funcionalidad implementada del monolito para el manejo de resultados de clasificación. Se utiliza dicho comportamiento para agregar lógica mínima que se encargue de la lectura del CSV. De esta manera a partir de las conductas presentes en el CSV (ya clasificado), se arma la serie de indicadores, almacenándola y modelándola en las estructuras internas del monolito. Dicha estructura interna permite obtener el JSON asociado a los perfiles generados de forma trivial.

4.3.4. Broker

Operación	Método HTTP	Parámetros	Salida
Clasificación con DL a partir de CSV	POST	Archivo ZIP con los CSV Cantidad mensajes Enfoque de redes	JSON con los perfiles generados
Clasificación con DL a partir de ARFF	POST	Archivo ZIP con los ARFF y el CSV de los timestamp Cantidad mensajes Enfoque de redes	JSON con los perfiles generados
Clasificación con DL a partir de Takeout	POST	Archivo ZIP con los JSON Enfoque de redes	JSON con los perfiles generados
Clasificación con DL a partir de BD	GET	URI de la BD Nombre de la BD Cantidad chats Cantidad mensajes Enfoque de redes	JSON con los perfiles generados
Obtención de Resultados	GET	Nombre de Tabla	JSON con los resultados de la tabla solicitada

Obtención de Resultados	GET	Nombre de Tabla Nombre Integrante	JSON con los resultados de la tabla solicitada para el integrante solicitado
Obtención de Resultados Disponibles	GET	(sin parámetros)	JSON con listado con el nombre de los resultados
Almacenar Resultados	POST	CSV con los resultados de la clasificación Nombre de tabla	Confirmación del alta realizada

Tabla 4.4. Exposición HTTP del servicio encargado de gestionar la interacción entre los servicios que proveen la funcionalidad de la aplicación web.

El broker es el encargado de funcionar como una única capa de abstracción para el cliente, para que no tenga que conocer a cada uno de los servicios necesarios para proveer la funcionalidad necesaria. Dicho concepto central de abstracción, indica que el broker debe ofrecerle al cliente toda la funcionalidad necesaria, es decir que debe englobar de cierta manera a todos los endpoints del resto de los servicios. De cierta manera porque puede que exista algún tipo de endpoint que sea utilizado por una herramienta especial para desarrollo, que no esté expuesta al usuario final. Igualmente, la funcionalidad ofrecida por el broker al cliente, equivale a la que ofrecen todos los servicios por separado.

Como se podrá observar en la Tabla 4.4, todas las operaciones son muy similares a las que realizan los servicios con los cuales el broker interactúa, aunque la diferencia se puede palpar en la salida de los endpoints del broker. Por ejemplo, para la clasificación de mensajes con técnicas de aprendizaje profundo, se vio que, en el servicio encargado de dicha tarea, se retorna un CSV. En este caso lo que se retorna es un JSON con los perfiles generados. Esto se debe a que se está analizando el broker, es decir que cuando el cliente le hace una solicitud, espera los perfiles como respuesta. Las peticiones que hace internamente el broker no es de interés para el cliente, ni toma conciencia de ello, lo cual hace visible la noción de abstracción que provee el broker al cliente.

4.3.5. Cliente

Este servicio no requiere mayores detalles en cuanto a los endpoints debido a que consiste en un servidor web con un solo endpoint, que se trata de un método GET sin parámetros que como respuesta envía los archivos correspondientes con la aplicación web a utilizar por el usuario. En cuanto a funcionalidad no posee mayores comentarios a destacar.

4.4. Tecnologías Empleadas

Hasta el momento se hizo mención a la funcionalidad que proveen los servicios, así como también a los endpoints que expone cada uno para acceder a dicha funcionalidad. Se explicó tanto el funcionamiento de cada servicio, como otros aspectos como por ejemplo la arquitectura de la aplicación web, en donde la existencia de un broker le provee abstracción al cliente en cuanto al funcionamiento del backend, con todos los beneficios (y también amenazas) que ello provee.

En esta sección se hará especial énfasis en los detalles concretos de la implementación, referido a las tecnologías empleadas. Se efectúa un listado de las tecnologías empleadas, junto con las características de cada una, para luego realizar un barrido por cada servicio en sección siguiente, detallando la forma en que cada tecnología se aplicó a cada uno de ellos. En secciones anteriores se evitó proveer mayores explicaciones sobre la tecnología empleada, a fin de hacer hincapié en la funcionalidad de cada uno, abstrayéndose dicha explicación de la tecnología subyacente que materialice los conceptos detallados.

Las tecnologías que se emplearon se corresponden a diversas áreas, entre las que se destacan:

- Web Services
- Data Science
- Deep Learning
- Bases de Datos
- Front End

A continuación, se seguirá el orden de dicho listado para detallar cada una de las tecnologías empleadas.

4.4.1. Spring Boot

Consiste en un framework basado en Java, que se utiliza para desarrollar servicios web. Se caracteriza por su facilidad de uso, en donde los pasos burocráticos que requiere para crear un proyecto, son realmente mínimos. Con respecto a pasos burocráticos se abarca todo tipo de tareas como por ejemplo las diversas configuraciones iniciales que se deben realizar, la disposición de los directorios, etc. Spring Boot[9] se caracteriza por automatizar casi todas esas tareas. Lo único que se necesita para tener una aplicación Spring corriendo es una selección inicial de las librerías a utilizar, que no se realiza necesariamente desde Maven o Gradle, sino desde lo que se denomina Spring Initializr[10], que consiste en una interfaz de usuario sencilla e intuitiva en donde al seleccionar las librerías a utilizar, permite descargar un proyecto ya generado, con las dependencias ya importadas.

[9] https://spring.io/projects/spring-boot

[10] https://start.spring.io/

Otra de las características distintivas de este framework, es el formato automático que se le da a los retornos de las funciones. Por ejemplo, en caso de anotar un determinado método como un endpoint, si dicho método retorna una estructura de la clase HashMap, Spring automáticamente enviará dicha información como si fuese un JSON, que contendrá un diccionario con los datos de la estructura retornada. Esta característica es sumamente potente, ya que no solo funciona con estructuras predefinidas por el lenguaje como por ejemplo el HashMap, sino que también se realiza un mapeo automático con las variables internas de la clase que se quiere retornar. Por ejemplo, en caso de tener la clase CuentaBancaria con las siguientes variables internas:

- private String titularCuenta
- private int saldo
- private List<String> historialMovimientos

En caso de que un método quiera retornar una lista de objetos de la clase CuentaBancaria, lo que se verá desde el cliente es un JSON, que contendrá un arreglo (por la lista) de diccionarios, donde cada diccionario tendrá tres pares clave-valor. Cada par tendrá el nombre de la variable como clave, y el valor concreto de la variable como valor asociado a la clave. A modo de graficar el ejemplo, se proporciona la Figura 4.7. Esta funcionalidad resulta ser de gran utilidad para el manejo de archivos JSON, muy comunes en arquitecturas REST. De esta manera, al tener dicha facilidad provista por Spring para la creación de archivos JSON, se toma como decisión de diseño la adopción de dicho formato.

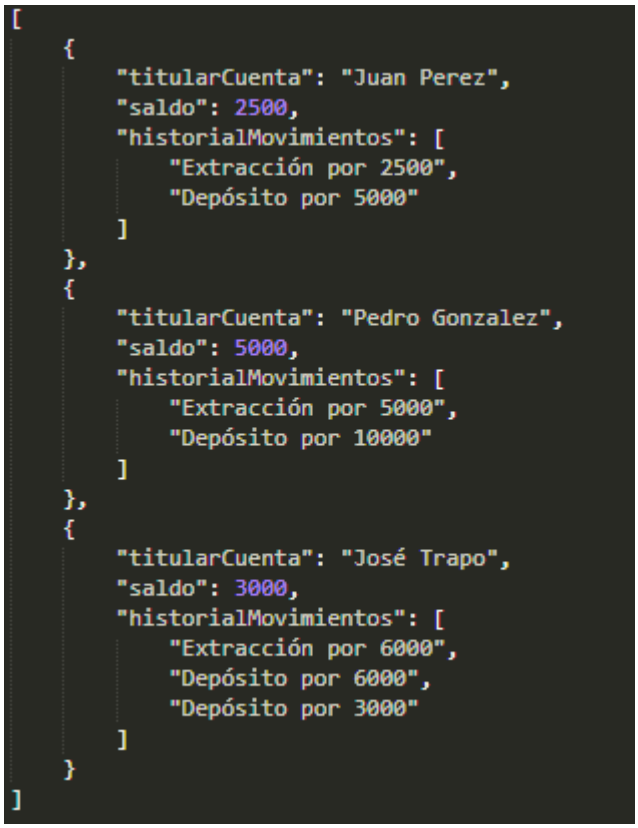

Figura 4.7. Ejemplo ilustrativo de mapeo automático de las clases por parte de Spring.

Cabe aclarar que para que dicho mapeo se lleve a cabo no es necesario realizar trabajo extra a la hora de implementar la clase, como por ejemplo recurrir a redefinición de método. Todo el mapeo se hace de forma automática sin necesidad de burocracia extra a realizar por parte del desarrollador.

Con respecto al desarrollo de aplicaciones REST, Spring Boot se maneja a través del empleo de anotaciones, las cuales, en las sucesivas iteraciones del Spring, se han ido puliendo, al punto de que la siguiente anotación "@PostMapping("/obtener_arff_csv")" indique que, sobre el método anotado, se establezca un endpoint que sea accedido a través de una petición POST sobre la URL especificada por parámetro. Luego se tienen otro tipo de anotaciones que funcionan sobre los parámetros del método anotado, como por ejemplo "@RequestBody MultipartFile zipFile". De esta manera se solicita que la petición POST incluya en su cuerpo un determinado archivo. Para el caso ejemplificado, se trata de un archivo ZIP con los ARFF dentro, para traducirlos a un CSV a emplear para su posterior clasificación.

4.4.2. Flask

Se trata de un framework basado en Python. A diferencia de Spring, Flask se autodenomina un microframework[11], ya que provee mucha menos funcionalidad que Spring. La ventaja que provee, es la velocidad de desarrollo debido al lenguaje que utiliza, que es Python. Este lenguaje se diferencia de Java principalmente por el tipado dinámico, es decir que Python es interpretado, a diferencia de Java, que es compilado. Una de las grandes ventajas que tiene Python en comparación con Java, es la verbosidad. Las características de Java hacen que se necesiten diversas implementaciones para que se tenga un programa funcionando. Por ejemplo, se necesita crear un proyecto, luego una clase, y dentro de dicha clase, definir el método principal, es decir aquél por donde inicia la ejecución. Toda esta serie de pasos en Python no es necesaria, se reduce a crear un archivo, escribir un determinado código, y con correrlo directo ya se encontrará funcionando. Además, no se deben escribir los tipos de las variables, a diferencia de Java, lo cual significa otro gran ahorro en términos de verbosidad del código. Puede darse una gran discusión e intercambio de ideas sobre qué lenguaje es mejor para desarrollar servicios web, comparando Java o Python, entre otros, pero va más allá del alcance de este trabajo de grado. Solo se limitará a explicar que la elección de cada tecnología se dio a fin de facilitar el reuso de la funcionalidad implementada. Igualmente, más adelante en esta sección se hará un repaso de por qué se utilizó cada tecnología en cada servicio, sin perder en el medio el foco del trabajo realizado.

Volviendo a la descripción de Flask, como se detalló anteriormente, al momento de implementar microservicios, es más rápido que Spring en cuanto a tiempos de desarrollo, aunque dicha ventaja se da más por las características de Python que por el framework en sí. En la Figura 4.8 se muestra un ejemplo de un endpoint desarrollado para el servicio de consulta de resultados, en donde se puede apreciar que la ausencia de tipos explícitos provoca que el código implementado sea mucho más compacto, en comparación con Java.

[11] http://flask.pocoo.org/

```python
@app.route('/<tablename>', methods=['POST'])
def add_conversation(tablename):
    csv_file = StringIO(request.files['csv_file'].stream.read().decode('utf-8'))

    load_classification_into_db(properties['db_file'], csv_file, tablename)

    return jsonify(mensaje='alta de tabla realizada exitosamente')
```

Figura 4.8. Ejemplo de endpoint desarrollado en Python, empleando Flask. El tipado dinámico que posee el lenguaje permite código más compacto, además de reducir los tiempos de desarrollo.

4.4.3. Pandas

Es una librería escrita como extensión de NumPy para utilizar con Python, que provee estructuras de datos y herramientas de análisis para esos datos. Dichas estructuras y herramientas se caracterizan por su facilidad de uso, en donde los conjuntos de datos con los cuales se quiere trabajar se modelan a través de una estructura básica llamada DataFrame. Dicha estructura consiste en una especie de tabla que en cuanto a apariencia se puede asemejar a una planilla de Excel, o a una tabla de SQL. En un DataFrame normalmente cada fila se corresponde con un ejemplo o instancia, mientras que cada columna modela a una característica de cada ejemplo. Partiendo de la estructura con la cual se trabaja, las herramientas que provee Pandas consisten en diversas operaciones sobre los DataFrames. Esta librería está optimizada en cuanto a rendimiento, ya que gran parte de la funcionalidad que provee no está escrita en código Python nativo, sino en lenguajes de más bajo nivel.

Dada su característica principal de trabajar con DataFrames, Pandas se lleva muy bien determinadas fuentes de información, como por ejemplo CSV ya que la estructura de ambos es altamente compatible. Lo mismo ocurre con SQL. Los DataFrames también se compatibilizan muy bien con las tablas de las bases de datos relacionales, por lo que la interacción entre dichas tecnologías es sumamente sencilla. Por ejemplo, la clase DataFrame provee un método llamado to_sql en donde la estructura, sin modificación alguna en su contenido, se almacena en la tabla cuyo nombre está parametrizado en dicho método. En este caso lo único que necesita el método es una conexión abierta con la base de datos. Para mostrar la simplicidad y lo bien que se integra Pandas con SQL, se provee en la Figura 4.9 un ejemplo de implementación en donde un CSV es cargado en memoria, es decir en un DataFrame, y luego dicha estructura se almacena en una base de datos.

```
def load_classification_into_db(db_file, csv_file, table_name):
    connection = sqlite3.connect(db_file)

    df = pd.read_csv(csv_file, sep=',')
    df.to_sql(table_name, connection, if_exists='replace')

    connection.commit()
    connection.close()
```

Figura 4.9. Ejemplo de la facilidad de integrar CSV con Pandas y SQL. A modo aclaratorio la variable "pd" hace referencia a Pandas, no se trata de ninguna otra variable predefinida.

Además del ejemplo expuesto, Pandas se ha convertido en un estándar para trabajar en el área de Data Science, debido a la facilidad con la cual permite que se haga manipulación de información. Y por carácter transitivo, también es muy utilizada para realizar tareas de aprendizaje automático, más precisamente al momento de realizar preprocesamiento del dataset. Cualquier tipo de operación que se quiera realizar con el dataset, en caso de efectuarla utilizando Python plano en vez de Pandas, tendría una complejidad exponencial. Generalmente lo que suele pasar cuando se inicia en el área de aprendizaje automático, es querer reinventar la rueda al querer realizar una determinada tarea sobre el dataset. Una vez que se atraviesa sobre dicho proceso, se empieza a descubrir Pandas, observando y palpando la diferencia entre reinventar la rueda y utilizar cosas que ya están hechas, y que dan mejores resultados.

4.4.4. Keras

Las redes neuronales, como se explicó en el background teórico, son estructuras de estadística aplicada en donde, al momento de su implementación hay dos formas de desarrollarlas. La primera es desde el punto de vista matemático, es decir especificando las operaciones matemáticas que se llevan a cabo, que posteriormente en su conjunto (sin incluir otros aspectos como por ejemplo la función de costo) constituyen la red neuronal propiamente dicha. En este enfoque de desarrollo, lo que se plantea es un grafo que modela un flujo de operaciones. Sin embargo, no es la única forma de desarrollar una red neuronal. Este enfoque recientemente descrito es más orientado a la matemática pura, entre otras cosas. Existen librerías que se encargan de ese tipo de desarrollo, como por ejemplo TensorFlow [40], que trabajan con redes neuronales, pero a más bajo nivel.

Por otro lado, se tiene otra la forma de desarrollar modelos de aprendizaje profundo, que se caracteriza por trabajar por encima del enfoque anterior. Es decir, trabaja sobre una capa de abstracción. Esta forma de trabajar es mucho más sencilla que la anterior, ya que no se trabaja con operaciones matemáticas, sino con capas o *layers*. En la Figura 4.10 se provee un ejemplo de red neuronal que conceptualmente es muy interesante, pero más interesante es la poca cantidad de líneas de código para crearla. En este caso se ejemplifica solamente el modelo, es decir que no se plantea la función de pérdida, ni el código correspondiente al entrenamiento de dicho modelo.

Todos los detalles relacionados al desarrollo de las redes neuronales aplicado al trabajo de grado, se discuten con especial profundidad en el Capítulo 5.

```
from keras.models import Sequential
from keras.layers import Input, Convolution2D, MaxPooling2D, Flatten, Dense, Dropout

model = Sequential()

X_input = Input(shape=(1, 28, 28))
X = Convolution2D(nb_col= 3, nb_filter=4, activation='relu' , nb_row=3)(X_input)
X = MaxPooling2D()(X)
X = Convolution2D(nb_col= 3, nb_filter=8, activation='relu' , nb_row=3)(X)
X = MaxPooling2D()(X)
X = Flatten()(X)
X = Dense(activation='relu' , output_dim=100)(X)
X = Dropout(p=0.4)(X)
X = Dense(activation='softmax' , output_dim=10)(X)

model = Model([X_input],[X])
```

Figura 4.10. Ejemplo de red neuronal convolucional empleando Keras. Como se puede observar, la utilización una librería que funcione como capa de abstracción puede simplificar y acelerar el desarrollo de los modelos predictivos basados en redes neuronales.

Como sucede en toda capa de abstracción, pueden surgir situaciones en donde el contexto del desarrollo de la red neuronal derive en casos donde se deban hacer cosas muy específicas, como por ejemplo una función de costo personalizada. En este tipo de casos Keras deja de ser la mejor opción para emplear, teniendo que optar por otras tecnologías destinadas a trabajar con las redes neuronales, pero más cerca de la operación matemática, como por ejemplo TensorFlow, MXNet [41], CNTK [42], Caffe [43], PyTorch [44], etc. Aunque en estos casos, la complejidad del desarrollo aumenta al tratarse de librerías más complejas, lo cual introduce un trade-off al trabajar con este tipo de problemáticas: facilidad de desarrollo vs. capacidad de control sobre el modelo.

4.4.5. SQLite

Se trata de una tecnología de gestión de bases de datos relacionales que se caracteriza por su tamaño reducido, por lo que su uso es muy extendido en aplicaciones móviles debido a su bajo consumo de recursos. La característica distintiva que posee SQLite es que, a diferencia de otros sistemas de gestión, SQLite no funciona como un proceso separado, es decir con su propio servidor. Esto quiere decir que se utiliza como una base de datos embebida, en donde la base de datos propiamente dicha, se almacena en un archivo con extensión ".db". Esta característica resulta fundamental para el desarrollo veloz de los servicios, ya que la conexión con la misma se reduce simplemente a la solicitud del archivo dentro del mismo directorio en que se está trabajando.

Otra característica que resulta de suma utilidad es la configuración de la misma. Comúnmente, en caso de utilizar bases de datos tradicionales, se debe instalar el servidor, conectarse a una determinada URL con un determinado usuario y contraseña, crear el esquema, y otro tipo de cosas. En el caso de SQLite, al momento de conectarse, es decir al querer conectar al archivo que contenga la base de datos, si el archivo en la ruta provista no existe, lo crea automáticamente, con un esquema preestablecido. Las tareas relacionadas a alta de tablas se podrán llevar a cabo perfectamente sin complicación alguna.

4.4.6. MongoDB

Se trata de una base de datos no relacional, en la que, a diferencia de sus contrapartidas relacionales, no se maneja los conceptos de tablas, restricciones de integridad, etc. El almacenamiento de la información se realiza a través de documentos, cuya apariencia es la de un JSON, pero con estructura dinámica, es decir que a través de una sentencia de actualización se pueden agregar más campos en un registro. La finalidad de MongoDB es no poseer las características ACID completas (propias de las bases de datos relacionales) a fin de proveer mayor flexibilidad y escalabilidad. Las bases de datos tradicionales, es decir las relacionales, garantizan dichas características, pero al trabajar con grandes cantidades de información se tornan inviables debido a sus limitaciones de escalabilidad.

4.4.7. Angular

Por el momento se mencionaron todas tecnologías que trabajan del lado del servidor, del backend. Para el desarrollo de una interfaz de usuario amena e intuitiva para el usuario final, se debe desarrollar un front-end para dar formato a la salida del backend. Esto se debe a que, como se mencionó anteriormente, un hipotético supervisor no puede observar los perfiles generados en forma de JSON. Angular se trata de un framework para hacer front-end, que trataba con HTML y CSS para estructurar la página web, y Typescript para implementar la lógica. La forma de trabajar se basa en la noción de componentes y servicios. Un componente consiste a grandes rasgos en un mecanismo de abstracción de código HTML representando una vista. Por ejemplo, al suponer que una sección de una página web contiene una determinada cantidad de líneas de código HTML, dicha sección se puede contener en un componente. De esta manera, en el HTML principal, al incluir el tag que identifica a dicho componente, lo que en realidad se hace es incluir todo el código HTML de la sección original.

El código HTML de cada componente va en su correspondiente archivo. Cada componente tiene un determinado código Typescript que implementa la lógica inherente a su componente. A través de identificadores especiales, y palabras reservadas en el código HTML, es posible asociar dicho código con el Typescript. Luego con respecto a la comunicación inter-componente, existen los servicios, que se tratan de clases Typescript que se inyectan[12] en los controladores de cada componente. Lo

[12] https://angular.io/guide/dependency-injection

que se recomienda a la hora de trabajar con este framework es no implementar lógica en los controladores de cada componente, sino emplear para ello los servicios correspondientes.

4.4.8. Docker

Con respecto al tema del deployment, en el trabajo de grado se empleó Docker, que consiste en una tecnología encargada de automatizar esta tarea. Su funcionamiento está basado en el concepto de independencia de plataforma, es decir, que cualquier aplicación que funcione en Docker, funcionará de igual manera en cualquier plataforma que soporte Docker. Básicamente Docker es la misma idea que Java, pero a nivel de sistema operativo.

Dentro de esta tecnología se tienen dos conceptos fundamentales: imagen y contenedor. Cada aplicación o servicio que se desarrolla, se despliega en una imagen. Luego cada imagen, se ejecuta sobre un contenedor. Dicho de otra manera, el contenedor es la plataforma sobre la cual corre la imagen. De esta manera, uno al desarrollar un servicio, construye la imagen asociada, y cualquier PaaS o IaaS que soporte Docker, es capaz de recibir esa imagen, y que la aplicación web contenido en aquella imagen, corra de igual manera que por ejemplo en el equipo en donde se efectuó el desarrollo.

Al momento de construir las imágenes, se debe proveer una determinada configuración para la misma. Como por ejemplo archivos auxiliares para el servicio, puertos a exponer, etc. Todas estas cuestiones se definen en un archivo de texto plano llamado Dockerfile, el cual posee una sintaxis específica. Luego al momento de construir la imagen, Docker se basa en la configuración especificada en el Dockerfile del directorio de donde se invoca al comando correspondiente.

El problema que puede surgir cuando se tienen varios servicios, es la necesidad de tener que levantar un contenedor para cada imagen por separado, lo cual constituye ser una tarea sumamente engorrosa, sobre todo en casos de arquitectura con cientos de microservicios. Para solucionar esta cuestión, y otras más como por ejemplo la comunicación entre los contenedores, existe una tecnología denominada Docker Compose, que consiste en una herramienta que se utiliza por línea de comandos, pero que permite correr una aplicación multi-contenedor definiendo un solo archivo, llamado docker-compose.yml. En este archivo, se definen todos los servicios a levantar al correr el comando “docker-compose up”, especificando la ruta al Dockerfile de cada imagen, entre otras cosas.

4.5. Aplicación de Tecnologías

A partir de la descripción de cada una de las tecnologías que se emplearon en el presente trabajo de grado, en esta sección se detalla la forma en que cada tecnología se aplicó en cada uno de los servicios desarrollados. Partiendo de la funcionalidad inherente a cada componente, se desarrolla la forma en que las características distintivas de cada tecnología se utilizaron en cada feature implementada.

4.5.1. Clasificación

Con respecto a las redes neuronales desarrolladas, la tecnología que se utilizó para el desarrollo de las mismas es Keras. Debido a la abstracción que provee, el problema a resolver no requirió elementos que Keras no provea, por lo que se pudo aprovechar plenamente la aceleración en el desarrollo de las redes neuronales, en comparación con TensorFlow, otra de las tecnologías comúnmente utilizadas para este tipo de desarrollos. Al iniciar el desarrollo del modelo predictivo con el lenguaje Python, se optó por seguir en la misma línea a la hora del desarrollo del servicio. De esta manera se empleó Flask para implementar los endpoints REST que den acceso a la clasificación de los archivos CSV.

Con respecto a la lectura y manipulación de los archivos CSV, se empleó Pandas, ya que el pasaje de CSV a DataFrame es totalmente trivial, es decir basta con una sola línea de código. De hecho, en el ejemplo expuesto en la Figura 4.10, se efectúa dicha operación, de leer un archivo CSV y mapearlo a un DataFrame.

Como se explicó anteriormente, dicho servicio no obtiene el DataFrame y a partir de dicha estructura la somete a los clasificadores. Antes de ellos se realiza un preprocesamiento de los mensajes a clasificar mediante un mapeo del documento (oración que caracteriza a un mensaje), a su correspondiente embeddings. A modo de recapitulación, la entrada del preprocesamiento es el documento, y la salida es un vector de longitud fija, es decir un embedding. Dicho vector se genera a través del mapeo de cada palabra con su correspondiente word-embedding. Luego se promedian los word-embeddings resultantes, y se obtiene un “sentence-embedding”, que identifica a una determinada oración.

Como la teoría de word-embeddings indica, las relaciones entre palabras textuales y su correspondiente vector, debe estar predefinida. Esto quiere decir que se debe contar con una estructura de antemano que permita obtener un determinado vector dada una cadena de texto. Para ello se emplearon word-embeddings desarrollados por Cristian Cardellino [45], que consiste en un corpus de alrededor de 1.5 billones de palabras, compiladores desde diferentes corpus y recursos disponibles en la web de forma abierta. El formato con el que se cuenta con dichos word-embeddings, es un archivo de texto, para el cual fue necesario efectuar lecturas del mismo a fin de mantenerlo en memoria. De esta manera, mantener el mismo en memoria permitiría que cada palabra fuese mapeada a su correspondiente embedding en tiempos de latencia aceptables. La cuestión es que debido a la cantidad de palabras con las que se cuenta, es muy elevada (lo cual es algo sumamente positivo), el tamaño del archivo de texto es también muy elevada (aprox. 2.8 GB), por lo que complica la tarea de trabajar con el archivo utilizando estructuras de memoria tradicionales (léase HashMap, MultiMap o similar).

Como solución al manejo eficiente del corpus de embeddings, se optó por emplear una base de datos embebida en el clasificador, utilizando SQLite. De esta manera, se tomó el archivo de texto plano para luego insertarlo en dicha base. Los beneficios que dicho procedimiento otorgó, en comparación a utilizar una estructura en memoria, fueron el ahorro de la carga inicial de los embeddings (cada vez que se inicia el servicio, la carga en memoria del archivo demoraba un tiempo considerable), y el tiempo de acceso del vector asociado para cada palabra. Para esto último se

empleó un índice de SQL para acceder más rápidamente a las palabras en cuestión, ya que la operación a realizar consiste en dada una palabra, obtener el vector asociado. A modo de detalle adicional, la tabla empleada para almacenar los embeddings consistió de tantas filas como palabras haya en el corpus, y dos columnas: la primera indicando la clave primaria, es decir la palabra concreta, y la segunda columna el vector de embeddings, almacenado en forma de BLOB, acrónimo de Binary Large Object.

Retomando el proceso anterior de preprocesamiento, se tiene que al querer mapear cada palabra con su correspondiente embedding, lo que se está realizando internamente, es un acceso a base de datos para cada palabra. Con respecto a la representación en memoria de cada embedding, el procedimiento a realizar fue utilizar arreglos de NumPy, debido a la eficiencia que provee dicha librería al realizar operaciones con vectores. Una vez que se tienen todos los embeddings asociados, se realiza un promedio entre ellos, resultando en un solo arreglo de NumPy. Este tipo de operaciones se realiza empleando operaciones propias de NumPy, lo cual hace que la tarea de hacer un promedio element-wise sea muy sencillo.

Una vez que se obtiene la representación asociada a un determinado documento, se somete al embedding al clasificador de reacciones. El clasificador de reacciones no es uno solo, ya que por parámetro del endpoint llega el enfoque de aprendizaje profundo que se quiera realizar. En este caso, los enfoques empleados son cuatro: uno con redes neuronales planas o densas, uno con convolucionales, otro con recurrentes, y por último un enfoque empleando convolucionales y recurrentes dentro de la misma arquitectura.

Una vez que a partir del enfoque que el usuario del servicio indicó, se seleccionan los clasificadores asociados. Esto es, por el momento un solo clasificador, para las reacciones. Luego en función de la reacción predicha para el embedding en cuestión, se selecciona un clasificador de conductas asociado a dicha reacción. Por ejemplo, en caso de que el primer clasificador indique que el mensaje posee una reacción negativa, en la segunda etapa se seleccionará un clasificador que realice predicciones para las conductas C10, C11 y C12, correspondientes a la reacción negativa. Finalmente, el resultado de la salida del clasificador de segunda instancia, constituye la salida final del modelo predictivo. El mensaje entró con el formato de cadena de texto, y salió con una conducta asociada.

Como se especificó anteriormente, el CSV a clasificar es modelado en memoria a través de Pandas, empleando un DataFrame. En cada línea de dicha estructura, se tiene una instancia, es decir un mensaje a clasificar. El proceso de clasificación se llevó a cabo agregando una columna más al DataFrame, correspondiente a la conducta predicha. De esta manera, cada línea del DataFrame fue mapeada a la misma línea, pero con un campo más, su conducta asociada. Una vez que se tiene la estructura con todas las conductas asociadas para cada mensaje, se utiliza la facilidad que Pandas ofrece para volcar un DataFrame en un CSV. Una vez generado dicho archivo, ya se va a tener generado el retorno del servicio.

4.5.2. Persistencia

La finalidad de este servicio es básicamente proveer persistencia para los resultados de las clasificaciones, para que el usuario final no recaiga en tener que recurrir a la clasificación cada vez que quiera generar perfiles. De esta manera se dispone de una base de datos que almacene los resultados de las sucesivas clasificaciones que un usuario realice a partir de sus fuentes de información. Los resultados de clasificación se proporcionan en formato CSV. Esto se debe a que la información de los indicadores que constituyen los perfiles de usuario, se muestra en forma de tablas, del lado del cliente.

Habiendo trabajado con SQLite y CSV durante la implementación del servicio de clasificación, y priorizando la velocidad de desarrolló, se optó por adoptar las mismas tecnologías para la implementación de este servicio. Con respeto a la disposición del servicio junto con sus endpoints, se utilizó Flask, un framework basado en Python. Como se detalló anteriormente, la facilidad de uso que Python y Flask otorgan, provoca que la calidad de la implementación se vea optimizada.

Para el desarrollo de este servicio se hizo uso de otra facilidad que Pandas ofrece con respecto al manejo de DataFrames. En este caso, el servicio recibe como entrada un archivo CSV, el cual se carga en memoria con la funcionalidad que Pandas provee. Luego la funcionalidad que se aprovecha para el desarrollo de este servicio es un método que permite guardar un DataFrame en una base de datos SQL. De esta manera el proceso de almacenar un CSV en un SQLite es sumamente trivial, como se expuso en la Figura 4.10, en donde el ejemplo que se expone en dicha imagen es justamente el método que se encarga de persistir un archivo CSV.

Cada clasificación se almacena en una tabla. Con respecto a las consultas, las mismas pueden ser de la tabla entera, o bien filtrando por usuario. De ésta última manera se pueden conseguir importantes ahorros en términos de tiempos de respuesta, lo cual impacta positivamente en la usabilidad de la aplicación web que consuma este servicio.

4.5.3. Gestión de Formatos y Generación de Perfiles

El monolito inicialmente fue desarrollado empleando Java plano, es decir sin utilizar ningún framework. Como originalmente fue pensado para funcionar como una aplicación de escritorio, no se realizó trabajo alguno relacionado a exposición de la funcionalidad. Es decir que la implementación desarrollada era para uso interno del monolito. Originalmente la idea del mismo fue utilizar el clasificador de texto de M. Mineo que utiliza técnicas de aprendizaje automático tradicional para clasificación de texto, siendo en este caso mensajes. La tecnología que emplea difiere bastante de la aplicación web desarrollada para el trabajo de grado. El clasificador original utilizaba como formato de representación de los mensajes, archivos ARFF. Este tipo de archivos proviene de Weka, una librería de Java destinada a tareas de minería de datos y aprendizaje automático [46]. Weka tiene sus propios clasificadores y, al igual que el framework desarrollado, trabaja con su propia fuente de información, aunque en este caso ARFF es un formato mucho menos común que CSV.

Además de las tareas de clasificación, el monolito desarrollado se encarga de realizar métricas y generar los perfiles de usuario, a partir de las clasificaciones obtenidas. Toda esta serie de funcionalidades que el monolito provee, fueron expuestas a través de modificaciones realizadas sobre dicho monolito. Consisten básicamente en la compatibilización de los accesos a diversos métodos, para que así puedan ser aprovechados al desarrollar posteriormente una aplicación de Spring Boot. De esta manera se implementaron los controladores necesarios para poder exponer la funcionalidad necesaria. Los perfiles de usuario generados, como se mencionó anteriormente, son en formato JSON, por lo que, recapitulando, los endpoint implementados para proveer esta funcionalidad, reciben un CSV correspondiente con la salida de las redes neuronales, es decir los mensajes clasificados. La salida es el susodicho JSON con los perfiles armados. Cabe destacar que, para la generación de los JSON, no se debió emplear librerías algunas, ya que dicha tarea es llevada a cabo de forma implícita por Spring Boot, como se explicó anteriormente en la Sección 4.4. de tecnologías empleadas.

En este caso la funcionalidad solicitada consistió en tres ítems. El primero radica en la generación de los perfiles, en donde se analiza la salida de la clasificación, en formato ARFF, y se la modela en memoria. Las modificaciones que se realizaron para esta funcionalidad es adaptar el modelado en memoria que se le da al resultado de las clasificaciones, para que pueda funcionar no solo con ARFF, sino también con CSV, el formato de salida de las redes neuronales.

Otra funcionalidad del monolito que se expuso mediante el uso de Spring Boot, que el manejo de fuentes de información. Como se expuso anteriormente, el tipo de información que el usuario final puede proveer para su posterior generación de perfiles empleando aprendizaje profundo es variado. A saber, ARFF, CSV, JSON (correspondiente a los Takeout), y acceso a base de datos. Estos tipos de fuentes de información son soportados por el monolito original, por lo que se tomó la decisión de aprovechar la funcionalidad implementada para el manejo de este tipo de formatos. Por lo tanto, se expuso como endpoint la funcionalidad correspondiente para convertir a formato CSV, para que así los mensajes provistos sean compatibles con el formato de entrada de las redes neuronales. En caso de que el usuario final provea un archivo CSV, no será necesario efectuar conversiones de formato, el broker reenvía dichos archivos directo a la red neuronal, sin tener que pasar en el medio por una conversión de formato, funcionalidad que se provee en este servicio.

A modo de resumen, la funcionalidad del monolito expuesta empleando el framework Spring Boot, es la siguiente:

- Conversión de formatos a CSV
- Generación de perfiles

4.5.4. Broker

Este servicio, como se mencionó anteriormente, es el encargado de interoperar con los distintos servicios desarrollados. En la Sección 4.2.4. se hizo referencia a las arquitecturas de servicios empleando brokers, comentando los pros y contras en el empleo de los mismos. En este caso, se desarrolló un único broker encargado y responsable de satisfacer la solicitud del cliente final. Esta tarea no la realiza por sí mismo, sino que lo hace a través de diversas llamadas HTTP a los distintos servicios con los que se cuenta. El broker fue implementado con Spring Boot junto con Unirest, una

novedosa librería Java para realizar peticiones HTTP livianas, caracterizándose por ser muy rápida y fácil de usar.

4.5.5. Aplicación Web

Con respecto al front-end, se decidió utilizar Angular, un conocido framework para este tipo de tareas. A partir de la implementación del broker, se tiene un cliente cuya finalidad es darle un formato más amigable con el usuario, a la información que retorna el broker. Es decir que el cliente se comunica únicamente a través del broker, abstrayéndose del resto de los servicios.

Dentro de Angular, se utilizaron componentes de Angular Material, cuyo diseño está inspirado en los lineamientos que Google establece, conocido como Material Design. Este tipo de componentes se caracterizan por el minimalismo en su diseño, priorizando un diseño limpio y conciso de la interfaz. La organización de la aplicación web se organizó empleando el concepto de Material Design llamado “tarjetas”, en donde cada una contiene cada feature de la aplicación. Luego con respecto a la visualización de los perfiles de usuarios se emplearon gráficos provistos por el paquete Chart.js[13], que consiste en una librería de código abierto que provee distintos tipos de gráficos desarrollados para utilizar en proyectos basados en JavaScript/Typescript. Estos gráficos se caracterizan por emplear también las directivas de diseño de Material Design, por lo que la integración visual de los gráficos con el resto de la aplicación web desarrollada se realiza de forma satisfactoria.

A continuación, en la Figura 4.11 se muestra una captura de la aplicación web implementada, en donde se puede ver que la funcionalidad que la misma ofrece está agrupada en las mencionadas tarjetas de Angular Material.

[13] https://github.com/chartjs/Chart.js

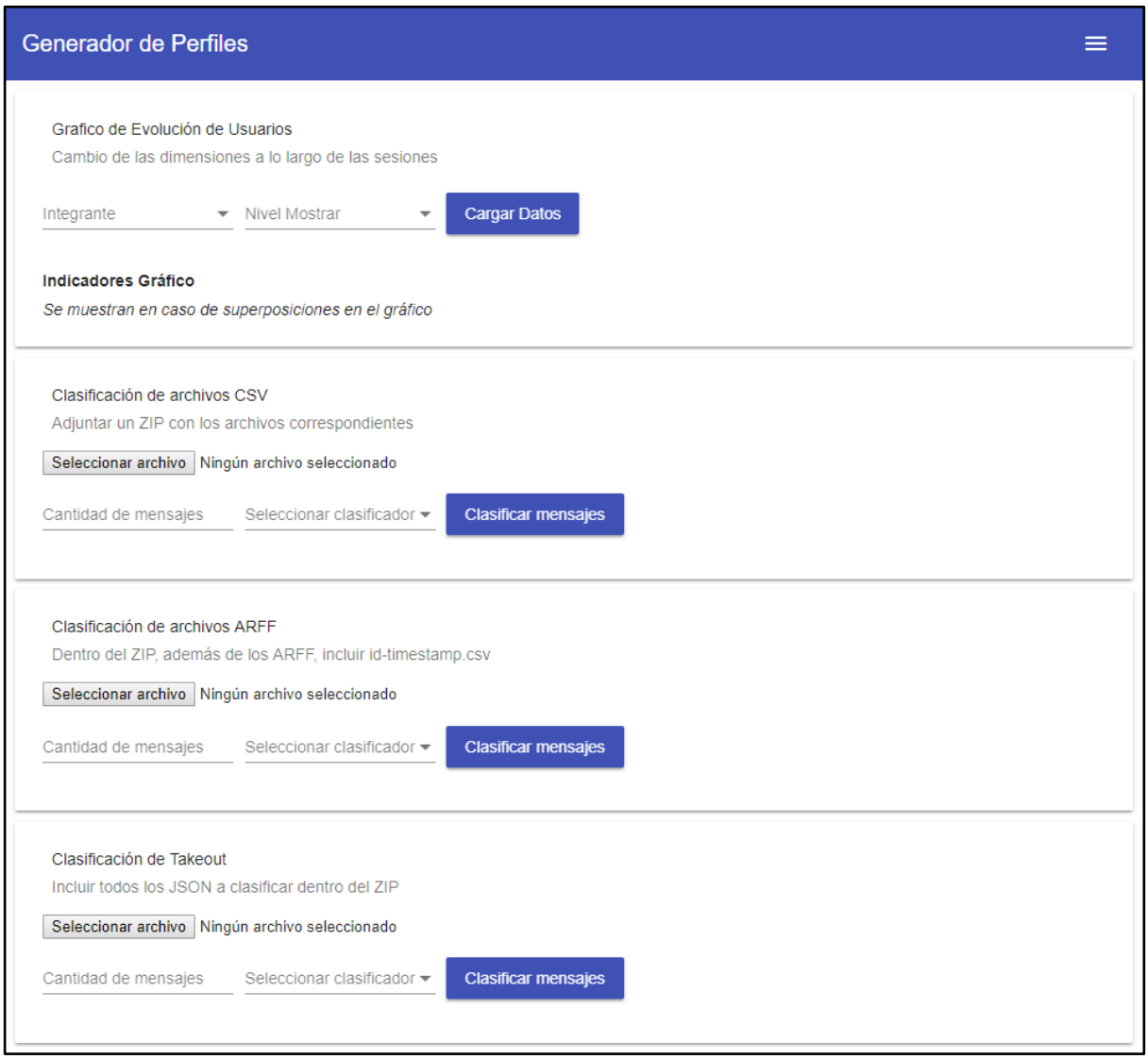

Figura 4.11. Apariencia de la aplicación web en donde se puede observar que la funcionalidad está contenida en tarjetas diferentes.

4.6. Escenarios de Calidad

A continuación, se detallan algunos escenarios de calidad considerados, a fin de relacionar la implementación llevada a cabo con los atributos de calidad de un sistema de software. El formato empleado es fuente-estímulo-respuesta.

4.6.1. Flexibilidad

Escenario de Calidad 1: Se desea que el sistema sea capaz de ser extendido fácilmente para trabajos futuros, sin que la arquitectura actual se vea afectada. Los diversos componentes que conforman al sistema deben estar débilmente acoplados a fin de que en caso de reemplazar y/o modificar alguno

de ellos, no se incurra en mayores modificaciones asociadas, que deriven de la modificación a realizar.

Fuente: Miembro de un equipo de desarrollo.

Estímulo: Modificar proceso predictivo.

Artefacto: Microservicio de clasificación.

Ambiente: Tiempo de implementación.

Respuesta: Localizar formatos de entrada/salida de información, funcionalidad modificada sólo en el servicio en cuestión.

Métrica Rta: Horas hombre ahorradas, horas del sistema fuera de servicio.

Para dar respuesta a este escenario, el diseño del sistema juega un papel primordial. El hecho de haber empleado una arquitectura orientada a microservicios permite que, en primer lugar, no sea necesario apagar el sistema entero para actualizar la funcionalidad del mismo. El procedimiento característico en sistemas de este tipo consiste en solamente reiniciar la ejecución del servicio afectado, siendo en este caso el correspondiente a la clasificación. De esta manera, cuando el broker envíe la información a la URL asociada con el microservicio de clasificación, la respuesta que se obtenga, provendrá de la nueva versión del modelo predictivo.

4.6.2. Escalabilidad

Escenario de Calidad 2: Se desea que el sistema pueda escalar correctamente al recibir una mayor cantidad de solicitudes de clasificación de mensajes. Debido a que el cuello de botella del flujo de la herramienta es la clasificación, cualquier variación que se pueda dar en la cantidad de usuarios simultáneos provocaría una importante alteración en los tiempos de respuesta del sistema desarrollado. Considerando una hipotética disposición del sistema en la nube, se desea que el sistema pueda escalar correctamente en la funcionalidad que lo necesite: en este caso la clasificación de los mensajes.

Fuente: Usuario Final.

Estímulo: Sucesivas solicitudes de clasificación de mensajes.

Artefacto: Sistema en producción.

Ambiente: Tiempo de ejecución.

Respuesta: El sistema es capaz de proveer mensajes clasificados en tiempos razonables para el usuario, en función de la cantidad de mensajes que se proporcionen.

Métrica Rta: Tiempo de respuesta del backend optimizado.

Este escenario de calidad fue satisfecho debido a dos factores: el primero relacionado a la arquitectura orientada a microservicios, y el segundo relacionado a Docker, la tecnología empleada para realizar el deployment de cada microservicio. El diseño del sistema, al separar la funcionalidad del mismo en microservicios autónomos, permite que, en caso de querer escalar una porción de la funcionalidad, solamente sea necesario escalar el microservicio afectado. Esto tiene especial relevancia en un contexto de deployment en la nube, en donde en caso de asignar más recursos a un servicio que no los necesita, los gastos asociados pueden crecer considerablemente. Con respecto a Docker, el manejo de contenedores, facilita el monitoreo de cada microservicio, lo cual se traduce en mejores datos que motivan las decisiones de escalar. El resultado es la posibilidad de escalar individualmente cada servicio, siendo en este caso el de clasificación, lo cual repercute directamente en menores tiempos de respuesta al momento de clasificar.

4.6.3. Usabilidad

Escenario de Calidad 3: Se desea que el sistema no necesite de diversas operaciones tediosas por parte del usuario, como por ejemplo la instalación del mismo en el equipo del usuario final. Tanto en el contexto de la herramienta como en el de las librerías asociadas. Una herramienta que requiera de diversas tareas burocráticas para su correcto funcionamiento, es una herramienta que mal predispone al usuario, por lo que es primordial realizar especial énfasis en esta cuestión

Fuente: Usuario final.

Estímulo: Acceso a la herramienta.

Artefacto: Sistema en producción.

Ambiente: Tiempo de ejecución.

Respuesta: Puesta a disposición de la aplicación de forma instantánea.

Métrica Rta: Tiempo transcurrido al acceder a la aplicación.

El diseño de la herramienta no se ideó como una aplicación de escritorio, ya que dicha decisión hubiese implicado diversas cuestiones como por ejemplo la necesidad de que el usuario final se deba encargar de descargar/instalar las tecnologías y/o librerías necesarias. La herramienta desarrollada en el presente trabajo de grado se ofrece en forma de aplicación web. De esta manera, el usuario final solamente debe contar con un navegador web, como por ejemplo Google Chrome o Microsoft Edge. Como la aplicación web accede a un backend remoto, los tiempos de acceso son mínimos, ya que consisten simplemente en la descarga de la página web asociada, por parte del navegador.

4.6.4. Portabilidad

Escenario de Calidad 4: Se desea que el sistema funcione correctamente independientemente de la plataforma subyacente. En caso de migraciones en el hosting del sistema implementado, garantizar que el correcto funcionamiento no se vea alterado repercute positivamente en el mantenimiento que se deba efectuar, lo cual se traduce en potenciales ahorros en los costos operativos.

Fuente: Miembro del equipo de operaciones.

Estímulo: Migración de hosting.

Artefacto: Sistema desarrollado.

Ambiente: Fase de deployment.

Respuesta: Migración exitosa con esfuerzo de tiempo mínimo.

Métrica Rta: Horas hombre para efectuar la migración.

El empleo de tecnología de contenedores provee independencia de plataforma. Esto quiere decir que en caso de que se testee el funcionamiento del sistema dentro de un contenedor en la misma máquina de desarrollo, dicho funcionamiento se verá inalterado independientemente de nodo que corra Docker. La única restricción en este caso es que el nodo debe tener Docker instalado, aunque en este escenario de calidad, dicha restricción es la que provee la solución. Al obtener la independencia de plataforma, no se deben destinar horas-hombre extra para la adaptación del servicio al nodo en donde se quiera correr, ya que Docker se encarga de esa tarea.

4.6.5. Interoperabilidad

Escenario de Calidad 5: Se desea que el sistema sea capaz de interactuar correctamente con servicios que constituyan trabajos futuros. La idea de plantear el presente trabajo de grado como un punto de partida para posteriores trabajos, requiere que la funcionalidad pueda ser accedida de forma sencilla por nuevos sistemas, a fin de facilitar el desarrollo de los mismos.

Fuente: Componente de un nuevo sistema.

Estímulo: Consulta de resultados al sistema desarrollado.

Artefacto: Sistema desarrollado.

Ambiente: Tiempo de desarrollo.

Respuesta: Los resultados pueden ser consultados fácilmente a través de la API asociada.

Métrica Rta: Horas-hombre destinadas a la comprensión de los medios de comunicación.

La arquitectura subyacente del sistema desarrollado, la cual está orientada a microservicios, implica un desacople total de cada uno de los microservicios que componen la funcionalidad. De esta manera, a través de una petición por HTTP que se le realice al servicio que almacena los resultados de clasificación, dichos resultados se podrán obtener fácilmente en formato JSON. Esto indica la

facilidad con la cual se opera con los servicios, ya que basta con conocer el endpoint al cual se le realiza la petición, para luego obtener la información asociada en un formato lo suficientemente estandarizado como lo es el JSON.

Capítulo 5: Enfoques de Deep Learning

En este capítulo se cubren todos los detalles relacionados al desarrollo de los modelos predictivos, abarcando en profundidad diversos temas como por ejemplo las arquitecturas de las redes neuronales empleadas, los hiperparámetros empleados para el entrenamiento de las mismas, etc. Se realiza este procedimiento para cada uno de los enfoques, contrastando las diversas capas empleadas y las métricas resultantes de los entrenamientos que se realizaron sobre cada arquitectura.

Si bien los modelos de aprendizaje profundo son más complejos de desarrollar que los de aprendizaje automático más convencionales, como por ejemplo redes bayesianas, siempre surge la disyuntiva entre qué tipo de clasificadores utilizar. La realidad indica que, si bien es correcto que el desarrollo de modelos predictivos basados en modelos de aprendizaje automático convencionales es más sencillo y, por lo tanto, más rápido, las métricas que obtienen pueden ser fácilmente superadas por los modelos de aprendizaje profundo. Generalmente, al momento de desarrollar modelos predictivos, lo que se recomienda es iterar rápidamente, según se ilustra en la Figura 5.1. Esto quiere decir: utilizar modelos que sean rápidos de implementar, para luego proceder a la experimentación y así obtener rápidamente un feedback que retroalimenta el desarrollo del modelo predictivo. Este proceso responde a la idea de que a veces el problema puede ser solucionado con modelos convencionales. Sin embargo, luego de varias iteraciones, es decir luego de varios ajustes sobre dichos modelos, las métricas ya no pueden mejorarse. En ese momento entran en juego las redes neuronales, que podrían considerarse como modelos más complejos, pero más “escalables” en caso de tratarse de un problema con más dificultad.

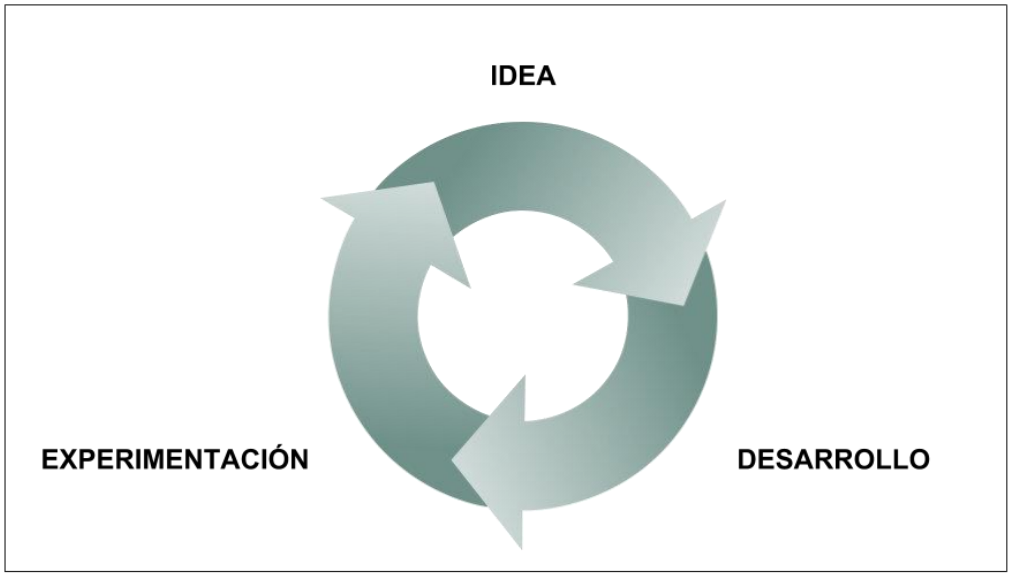

Figura 5.1. Metodología utilizada para el desarrollo de modelos predictivos. La recomendación es ciclar lo más rápido posible a fin de hacer crecer de forma proporcional, la base de conocimiento resultante de cada ciclo.

Una vez que se alcanzan los límites con los modelos más sencillos, es decir, una vez que las métricas no puedan mejorarse por más afinamiento que se haga sobre los hiperparámetros, se pasa a las redes neuronales. En el caso del problema que se quiere resolver, de clasificación de texto, se sobrepasa el límite del aprendizaje automático convencional para experimentar con diversas arquitecturas de redes neuronales. El objetivo claramente es maximizar las métricas a través de redes neuronales densas, convolucionales y recurrentes, analizando cómo se comporta cada tipo.

Cada tipo de red neuronal, constituye un enfoque distinto para realizar la clasificación de texto. A partir de cada enfoque, se realizan posteriormente diversos análisis comparativos y métricas. El fin consiste en el impacto de cada arquitectura en el proceso predictivo. Esto es, costo computacional, tamaño de las redes, matrices de confusión, etc. En la aplicación web que utiliza el usuario final, se provee la posibilidad de obtener el modelado de cada individuo a partir de cualquiera de los enfoques desarrollados.

A continuación, se realizará un barrido sobre cada uno de los enfoques, detallando para cada uno la arquitectura empleada, sus hiperparámetros, y sus curvas de precisión y pérdida, tanto para el dataset de entrenamiento como el de validación. Junto con cada aspecto de la información inherente a cada enfoque, se efectúa una conclusión sobre las decisiones tomadas. En este capítulo se detallan y discuten aspectos relacionados durante el entrenamiento de cada red neuronal. Es decir que no se incluye información alguna que tenga relación a la experimentación concreta, ya que se abordará en profundidad en el Capítulo 6, dedicado exclusivamente al análisis de resultados obtenidos.

5.1. Redes Neuronales Planas

Este tipo de modelos son los más básicos, ya que no se recurre a nada más allá de las típicas capas de redes densas o planas empleadas para realizar la clasificación propiamente dicha. En enfoques más complejos se recurre a otro tipo de conceptos como por ejemplo la convolución, recurrencia, o mezclas de ellos. Dicho de otra manera, en este enfoque se recae sobre la segunda etapa de la clasificación, como indica la Figura 3.9 del Capítulo 3. En dicha figura se muestra que generalmente las arquitecturas de redes neuronales se pueden dividir en dos etapas o secciones. La primera emplea la extracción de las características, que es donde se aplica convolucionales o recurrentes. En la segunda etapa o sección se realiza la clasificación concreta, a partir de las características extraídas durante la primer etapa. Esta segunda etapa de clasificación se realiza con redes neuronales planas

Volviendo al enfoque empleado, se usa uno básico, el cual no quiere decir que sea menos efectivo. En la Figura 5.2 se puede observar la arquitectura de la red. Como se puede observar, no hay presencia de recurrencia ni de convoluciones, aunque se hace énfasis en las técnicas de regularización para reducir el overfitting, un problema muy común en aprendizaje automático, que se acentúa en aprendizaje profundo.

Con respecto a la elección de las funciones de activación, lo que generalmente se suele recomendar es utilizar la función de rectificación (en inglés ReLU por Rectifier Linear Unit), en lugar de otras como la sigmoide o de tangente hiperbólica. Esta decisión radica en que el problema que tienen la

función sigmoide y la hiperbólica es que, para valores muy grandes o muy chicos, la pendiente de la función se pone muy plana, lo cual deriva en gradientes descendientes más lentos, lo que se traduce en peores resultados. La función de rectificación aprende mucho más rápido debido a que su derivada es cero cuando los valores que le llegan son menores o igual a cero. Por lo tanto, la recomendación en cuanto a funciones de activación, es utilizar siempre ReLU, y para la última capa, utilizar la sigmoide o Softmax en función de la cantidad de categorías que se quieran predecir, como lo es el caso de las arquitecturas que se utilizan en cada enfoque propuesto.

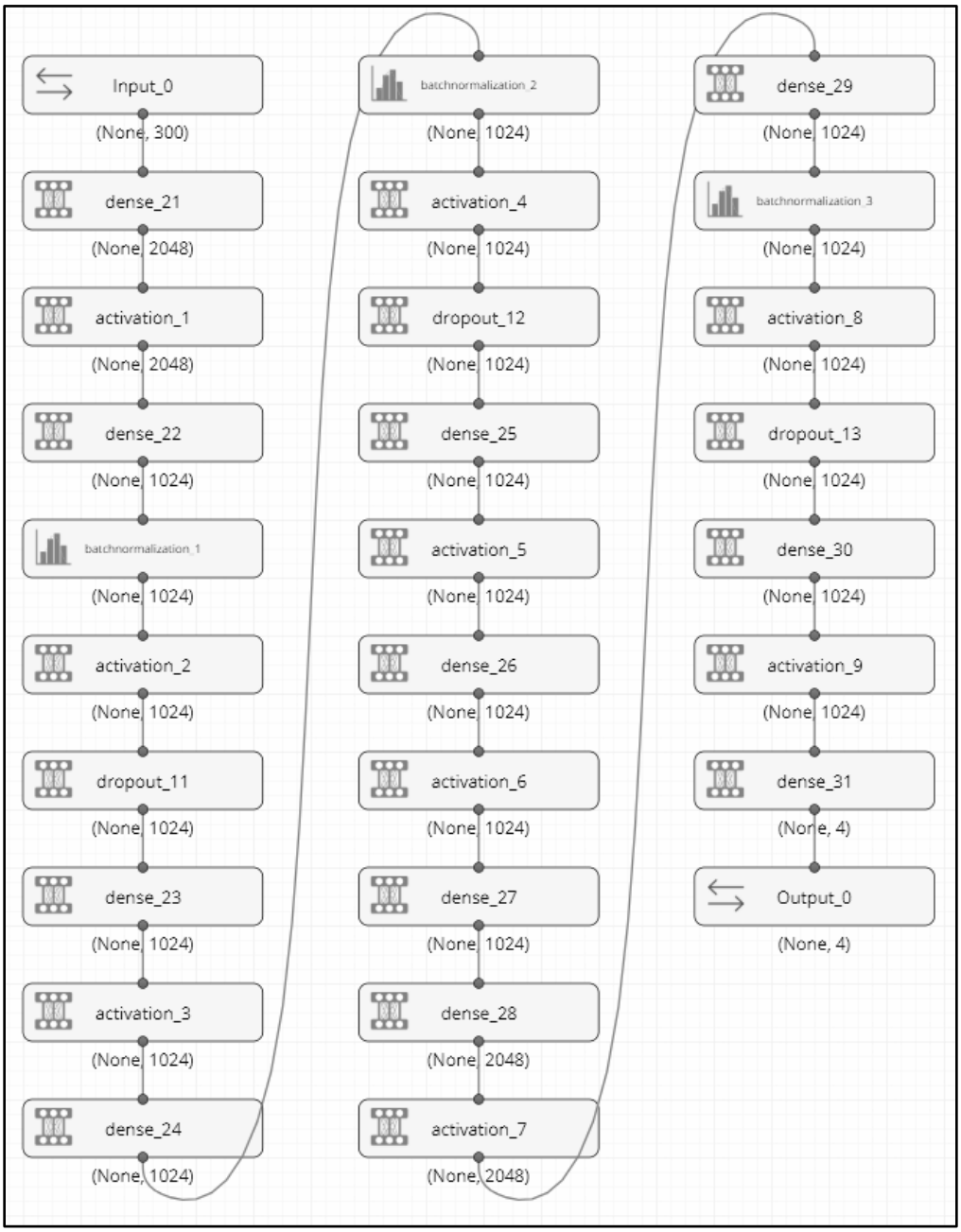

Figura 5.2. Arquitectura del primer enfoque empleando redes neuronales densas. Los números asociados al nombre de cada capa se corresponden con el nombre interno que se le dio a cada capa, no tienen ninguna injerencia en el funcionamiento de la misma.

Con respecto a la prevención del overfitting, se recurre una técnica de regularización la cual se ha probado ser muy eficaz. Se aplica Dropout sobre las salidas de cada capa densa, con un ratio de anulación de 0.6 que, si bien resulta ser un tanto elevado, ha resultado ser el óptimo durante el entrenamiento del modelo. Luego con respecto al escalado de las entradas de cada red, se utilizaron capas de normalización.

Como se puede observar, otro de los patrones de diseño que se suele seguir para desarrollar redes neuronales, es utilizar una cantidad decreciente de unidades en cada capa. En este enfoque si bien se usa la misma cantidad en las sucesivas capas, dicha cantidad en ningún momento crece, se mantiene igual o decrece. Este patrón constituye otra de tantas recomendaciones que se suele hacer durante el desarrollo de estos modelos.

Para el entrenamiento de este enfoque, los parámetros que se utilizaron fueron los siguientes:

Epochs	50
Tamaño del Batch	32
Función de Pérdida	Categorical Cross-Entropy
Optimizador	Adadelta

Tabla 5.1. Hiperparámetros utilizados para el entrenamiento del enfoque más básico.

Con respecto a las métricas obtenidas durante el entrenamiento del modelo, se proporciona la Figura 5.3 en donde se observan los gráficos de accuracy y pérdida, tanto para el dataset de entrenamiento como de validación. Dicha figura consiste en una función en donde en el eje Y se encuentra la métrica. Luego en el eje X, durante el entrenamiento, se muestra el batch por donde se encuentra el entrenamiento. Esto se debe a que en cada epoch (pasada por el dataset), se toma el dataset de a *batches*, es decir de a porciones. Como se podrá observar, en validación los números que muestra el eje X son distintos. Esto se debe a que la validación se efectúa al final de cada epoch, por lo tanto, lo que muestra es el número de epoch.

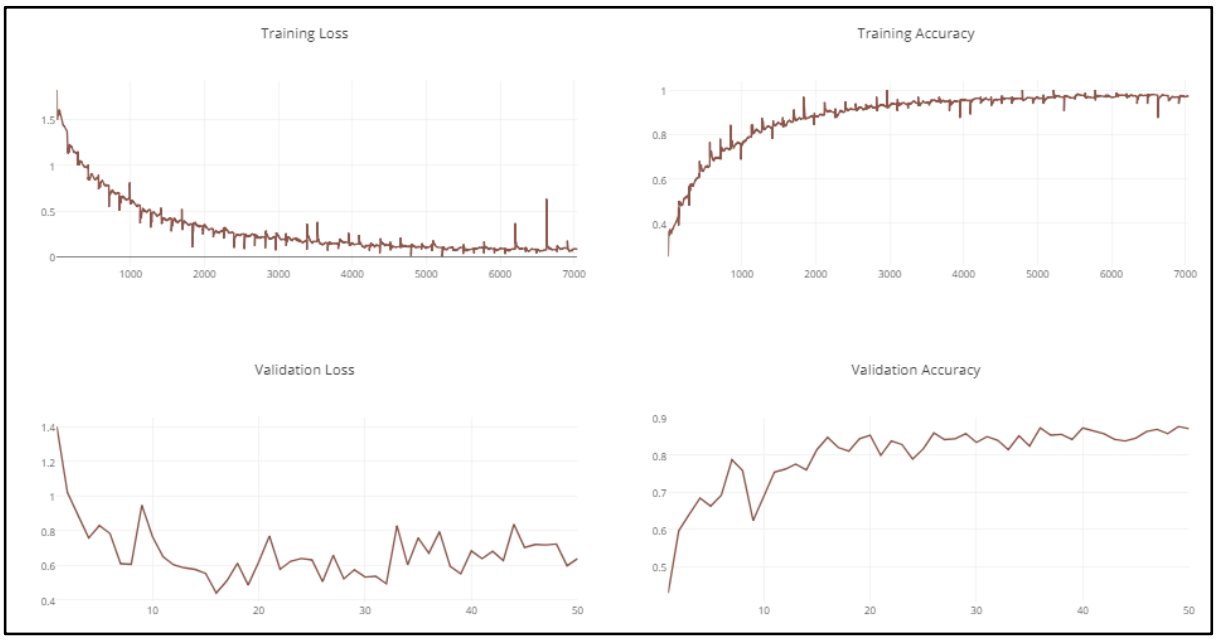

Figura 5.3. Métricas resultantes del entrenamiento del enfoque más básico.

5.2. Redes Neuronales Convolucionales (CNN)

Este segundo enfoque se emplean no solo redes densas para efectuar la clasificación, sino que, a diferencia del primer enfoque, se emplean redes que realizan procesamiento adicional para la *selección de características*. Dicho proceso de selección consiste en el descubrimiento automático de patrones en la entrada (en este caso un embedding), por parte de la red neuronal empleada. Para realizar este tipo de tarea, las redes densas corrientes no son la mejor elección, sino otro tipo de redes, llamadas convolucionales.

Como se explicó en el Capítulo 2 relacionado al background teórico, la operación de convolución consiste en aplicar un determinado filtro sobre una entrada. Estos filtros se aplican a fin de efectuar una determinada operación sobre la imagen, como por ejemplo invertir los colores de la misma, convertirla a escala de grises, etc. Además, estos filtros pueden ser capaces de detectar o reconocer determinadas formas en una imagen, a partir de un proceso de entrenamiento de estos filtros.

Si bien intuitivamente se infiere que la noción de convolución se aplica para imágenes, debido a su naturaleza multidimensional, la realidad indica que el concepto de convolución puede aplicarse a distintas cantidades de dimensiones, no solo dos o tres, como es el caso de una imagen. En este enfoque se experimenta aplicando la operación de convolución en los embeddings, por lo tanto, la operación de convolución que se emplea es unidimensional. De esta manera, se intenta descubrir patrones en las entradas, que no hayan podido ser descubiertos por las redes neuronales planas, es decir la del primer enfoque.

Como se puede observar en la Figura 5.4, se expone la arquitectura empleada para resolver el problema original, que concretamente es la predicción de una determinada categoría dado un embedding. En el Capítulo 3, se muestra en la Figura 3.9 que las arquitecturas de redes neuronales se suelen dividir en dos secciones. Mientras la primera se encarga de la selección de características recientemente mencionada, la segunda se encarga de la clasificación propiamente dicha. Además de la finalidad, ambas secciones se diferencian del tipo de unidades que emplean. La sección de clasificación, es decir la segunda, emplea siempre unidades densas, ya que han demostrado ser las que mejor funcionan. Dichas unidades también emplean sus correspondientes métodos de regularización, pero las funciones de activación se aplican siempre sobre unidades densas.

En la primer sección como se podrá observar, esto cambia. No se emplean unidades densas sino teoría más específica para la selección de características. En este caso se aplica la operación de convolución, aunque se puede recurrir a otros métodos, que serán detallados en los enfoques siguientes.

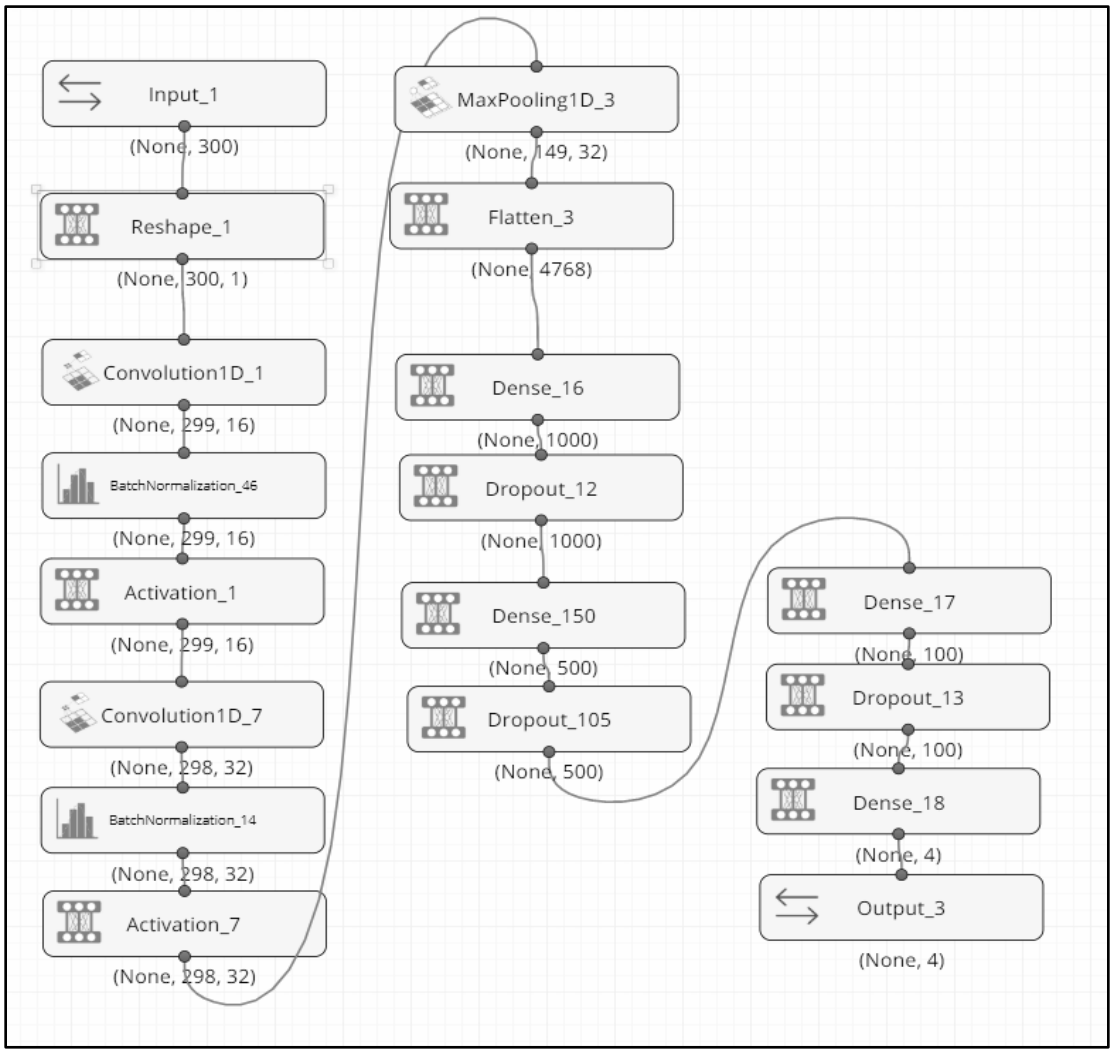

Figura 5.4. Arquitectura del segundo enfoque empleando redes neuronales convolucionales. Los números asociados al nombre de cada capa se corresponden con el nombre interno que se le dio a cada capa, no tienen ninguna injerencia en el funcionamiento de la misma.

Habiendo recapitulado el concepto de secciones dentro de una arquitectura de redes neuronales, en la Figura 5.4 se puede observar claramente la división entre cada una. Las unidades densas trabajan con tensores unidimensionales, por lo que es necesario que la salida de la primer sección sea sometida a una capa que “aplane” al tensor, a fin de compatibilizarlo con las unidades densas. Dicha capa constituye el punto en donde se aprecia la división entre cada sección.

Con respecto a los hiperparámetros de la arquitectura, las funciones de activación empleadas siguen el mismo lineamiento que en el enfoque anterior. Esto quiere decir que todas las funciones escogidas consisten en la función de rectificación (ReLU), mientras que para la última unidad se usa una softmax para la obtención de las distintas categorías posibles. Los hiperparámetros relacionados a las unidades convolucionales se pueden inferir a partir de la Figura 5.3. La primera unidad convolucional posee 16 filtros mientras que la segunda 32. Luego a partir de las dimensiones que se muestran, se puede inferir el tamaño de la ventana o filtro. Reduciéndose la primer dimensión de la entrada de 300 a 299, se puede concluir que la longitud del filtro es de 2. Como se trata de convoluciones unidimensionales, el tamaño del filtro también es unidimensional.

Luego se emplea otro patrón comúnmente empleado al momento de desarrollar redes neuronales convolucionales es el uso de capas de pooling luego del empleo de una unidad convolucional. Este tipo de decisión reside en varias razones. La primera es la complejidad computacional de este tipo de redes. Una de las características de la convolución es la cantidad de parámetros que emplea, debido a que, para obtener mejores métricas, se emplean varios filtros, lo cual multiplica la cantidad de parámetros, y por consiguiente la dificultad en el entrenamiento de las redes. Las capas de pooling contribuyen a esta cuestión al reducir la cantidad de parámetros, y por lo tanto trabajan como un regularizador, lo cual constituye la segunda razón que motive el uso de este tipo de capas.

A continuación, se tienen las capas de normalización, cuya finalidad es la misma que para el enfoque anterior, por lo que no amerita mayores detalles sobre las mismas. Con respecto a los hiperparámetros de las capas de Dropout, se obtuvieron las mejores métricas empleando el mismo ratio que, en el enfoque anterior, es decir de 0.6. Los hiperparámetros correspondientes al período de entrenamiento de la red, se exponen a continuación en la Tabla 5.2.

Epochs	100
Tamaño del Batch	32
Función de Pérdida	Categorical Cross-Entropy
Optimizador	Adadelta

Tabla 5.2. Hiperparámetros utilizados para el entrenamiento del enfoque basado en redes convolucionales.

Por último, con respecto a este enfoque, se adjuntan en la Figura 5.5. las métricas obtenidas durante el entrenamiento de la red. A diferencia del enfoque anterior, la cantidad de pasadas sobre el dataset se duplicó, debido a que, en este caso, la regularización que se efectuó fue más fuerte, por lo que se necesitó entrenar la red por más tiempo a fin de obtener mejores valores, sobre todo en

el accuracy y en la pérdida para el dataset de validación. En el enfoque anterior se debió entrenar por menos tiempo debido a que la función de pérdida se empezaba a disparar, lo cual indica una disminución importante en la capacidad de generalización de la red. Si bien en este caso se obtuvieron peores valores durante el entrenamiento, durante la validación se obtuvieron mejores resultados, lo cual inclina la balanza a favor de este enfoque, debido a la naturaleza del dataset de validación.

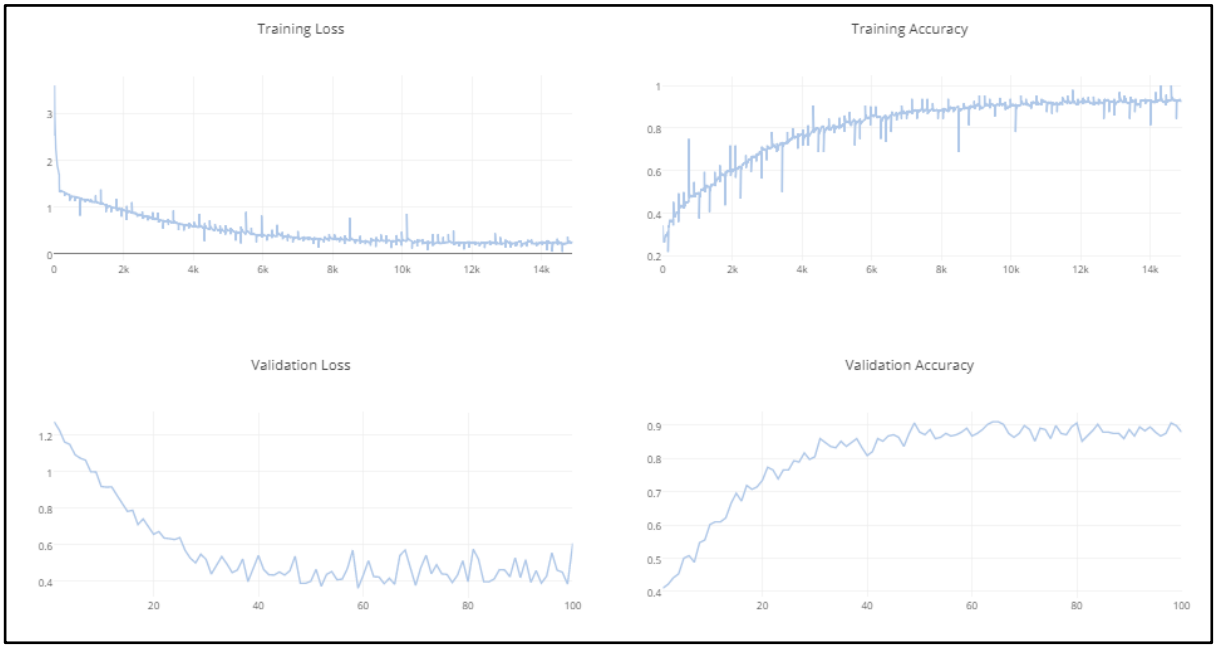

Figura 5.5. Métricas resultantes del entrenamiento del enfoque de redes neuronales convolucionales.

5.3. Redes Neuronales Recurrentes (RNN)

En este tercer enfoque, se emplea una arquitectura similar que en el enfoque anterior. Es decir que se tienen dos secciones, la primera especializada en la selección o detección de características en los datos de entrada, y la segunda especializada en la clasificación. La diferencia en este enfoque no radica en la segunda sección, sino en la primera. Se parte del concepto de la existencia de relación de dependencia dentro de una misma instancia, es decir de un mismo ejemplo. En enfoques anteriores, se consideraba a cada feature de la entrada de forma independiente, o sea que no se tomaba en cuenta que el valor de una feature pudiera influir en el valor de otra.

Para poder contemplar este tipo de situaciones hace falta la existencia de una noción de persistencia dentro de cada unidad, lo cual es resuelto por las redes recurrentes. En este enfoque, se plantea una arquitectura en donde la selección de características se produce empleando unidades LSTM (Long Short-Term Memory). Este tipo de problemas suele resolverse con dos tipos de unidades: las LSTM y las GRU. La diferencia entre ambas es la complejidad de cómputo de cada una. Las LSTM son más complejas y tienen más facilidad para aprender, aunque para grandes arquitecturas de redes neuronales complican mucho el entrenamiento, es decir que se podría decir que no son las mejores

en términos de escabilidad. En ese tipo de casos, se utilizan las GRU, que consisten en unidades más simples, que tardan más en aprender, pero hacen viable el entrenamiento de las redes neuronales de mayores dimensiones. En este caso, el tamaño del problema a resolver ameritaba el uso de LSTM, por lo que se concluyó utilizar este tipo de unidades.

Con respecto al tema de hiperparámetros de la red, se puede observar en la Figura 5.6, que la cantidad de unidades LSTM es 300, y la cantidad de unidades densas es 100. Para efectuar la regularización correspondiente, se utilizó un ratio de Dropout mucho menor en comparación con los enfoques anteriores, siendo en este de caso de 0.3 para la salida de la capa de LSTM, y 0.4 para las unidades densas. Las funciones de activación siguen un lineamiento similar que en los enfoques anteriores. Se utiliza softmax para la activación final, ReLU para la densa, pero para la LSTM la función comúnmente utilizada es la tanh.

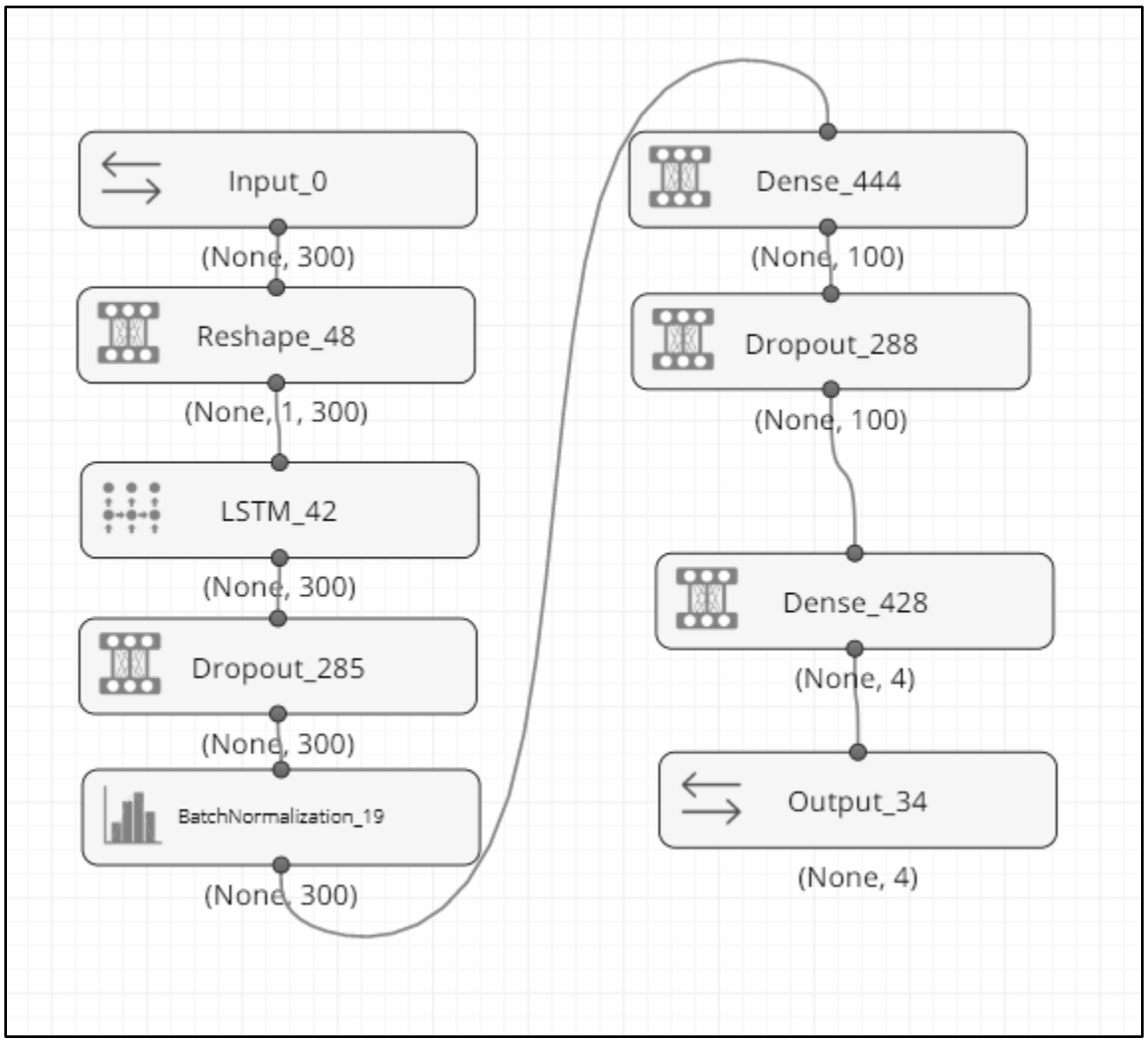

Figura 5.6. Arquitectura del tercer enfoque empleando redes neuronales recurrentes. Los números asociados al nombre de cada capa se corresponden con el nombre interno que se le dio a cada capa, no tienen ninguna injerencia en el funcionamiento de la misma.

Luego con respecto a los hiperparámetros utilizados durante el entrenamiento, se provee la Tabla 5.3 para mencionarlos. Como se puede observar, en este caso se pueden apreciar dos diferencias con respecto a los enfoques anteriores: la cantidad de pasadas, y el optimizador empleado. Las redes recurrentes son muy complejas para ser entrenadas, debido a las dependencias internas por la activación de cada feature que actualiza la celda de memoria de las LSTM. Esto provoca que, como dato al margen, las redes recurrentes sean inviables para ser entrenadas en CPU. Además de la dificultad que conllevan, demoran mucho más para converger, es decir que requieren más pasadas para que se puedan ajustar correctamente al dataset de entrenamiento, lo cual explica el uso de las 200 pasadas. Luego a través de los diversos entrenamientos realizados sobre esta arquitectura, se encontró que Adam es el optimizador que mejores resultados arrojó, estando muy por encima que el comúnmente utilizado, el Adadelta.

Epochs	200
Tamaño del Batch	32
Función de Pérdida	Categorical Cross-Entropy
Optimizador	Adam

Tabla 5.3. Hiperparámetros utilizados para el entrenamiento del enfoque basado en redes recurrentes.

Las métricas obtenidas en este enfoque demostraron una clara mejoría con respecto al enfoque anterior, de redes convolucionales. Como bien indica la Figura 5.7, los valores de precisión y pérdida son similares a los obtenidos en el segundo enfoque. Sin embargo, la clave de la mejoría estuvo en la función de pérdida para el dataset de validación, en donde para algunas pasadas se obtuvieron resultados de hasta la mitad. Por ejemplo, en donde el enfoque de convolucionales arrojaba una pérdida de 0.50, en este enfoque de recurrentes se obtuvo 0.25. Con respecto al valor de precisión para el dataset de validación, los resultados obtenidos fueron relativamente similares en comparación con el enfoque anterior, con sensibles mejoras en este enfoque.

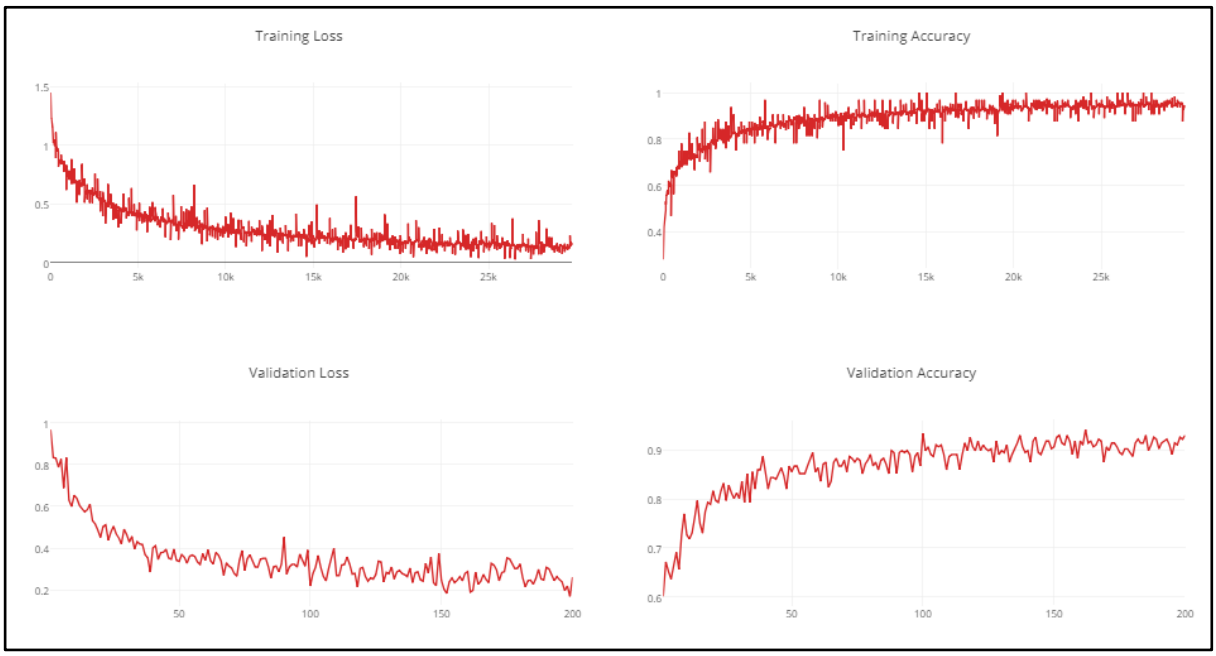

Figura 5.7. Métricas resultantes del entrenamiento del enfoque de redes neuronales recurrentes.

5.4. Redes Neuronales Convolucionales Recurrentes (CRNN)

Los resultados que arrojó el tercer enfoque son lo suficientemente satisfactorios como para usarlo en producción. Sin embargo, existe otro tipo de arquitecturas de redes neuronales en donde se aprovechan las características que proveen las redes convolucionales y las recurrentes. Dicha idea consiste en la posibilidad de que haya dependencias dentro de la misma instancia, pero habiendo estado procesada o filtrada previamente por la operación de convolución. Generalmente el patrón que se suele seguir, a partir de arquitecturas probadas que combinen estos conceptos, es aplicar primero capas convolucionales y luego recurrentes. Con respecto a la segunda sección, la misma se mantiene intacta, es decir que se mantiene el uso de unidades densas para efectuar la clasificación propiamente dicha.

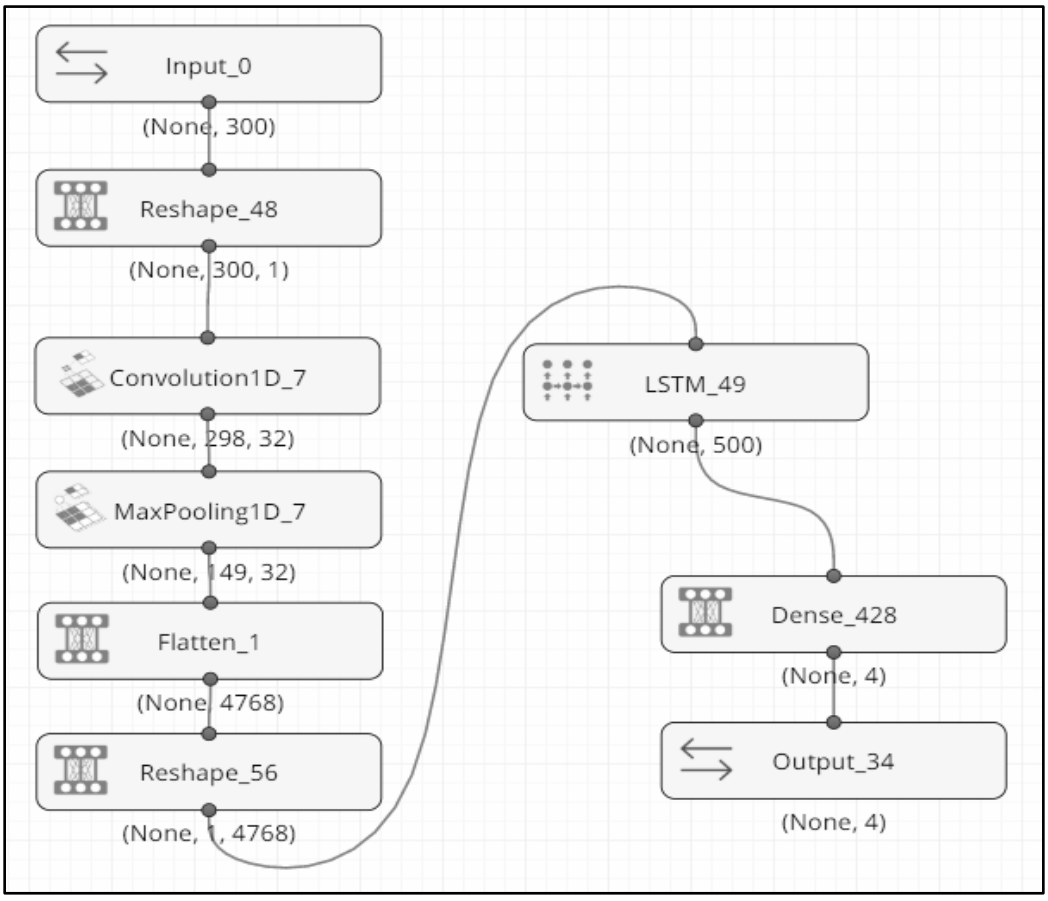

Figura 5.8. Arquitectura del cuarto enfoque empleando redes neuronales convolucionales recurrentes. Los números asociados al nombre de cada capa se corresponden con el nombre interno que se le dio a cada capa, no tienen ninguna injerencia en el funcionamiento de la misma.

Como se puede observar en la Figura 5.8, se empleó este patrón para el diseño de la arquitectura de la red neuronal. La sección de selección de características está compuesta por una capa convolucional junto con su correspondiente cada de pooling. Luego se emplea una capa de unidades LSTM para la detección de dependencias internas dentro de una misma instancia procesada por la capa convolucional. Un detalle a destacar al momento del desarrollo de arquitecturas de redes neuronales, es que el tamaño de la misma no guarda correlación con las métricas que produce la misma. Por ejemplo, en el caso de la experimentación con los enfoques propuestos, ha dado mejor resultado usar más unidades recurrentes por capas, en vez de utilizar varias capas de unidades recurrentes. Esta misma idea se corresponde con las capas convolucionales.

Con respecto a los hiperparámetros empleados en la arquitectura de la red, se pueden realizar diversos comentarios. Las funciones de activación siguen los mismos lineamientos, es decir que la capa convolucional utiliza la función de rectificación, la recurrente emplea una tanh, y para la clasificación final, una softmax. Para la capa convolucional, los mejores resultados se obtuvieron utilizando, como se puede observar en la arquitectura, 32 filtros, con una longitud de 3 para cada uno de ellos. Con respecto a la capa recurrentes, se utilizaron 500 unidades LSTM, que demostró

ser más efectivo en comparación con otros enfoques de emplear varias capas recurrentes con menos unidades. Luego se tienen capas auxiliares de cambios de dimensionalidad a fin de compatibilizar los tensores con las entradas de capas sucesivas.

Luego se tiene la Tabla 5.4, en donde se muestran los hiperparámetros empleados al momento del entrenamiento. Como se puede observar en dicha tabla, los mismos no variaron con respecto al enfoque anterior de redes recurrentes.

Epochs	200
Tamaño del Batch	32
Función de Pérdida	Categorical Cross-Entropy
Optimizador	Adam

Tabla 5.4. Hiperparámetros utilizados para el entrenamiento del enfoque basado en redes convolucionales recurrentes.

Los resultados obtenidos fueron ciertamente llamativos debido a la escasa expectativa de mejora con respecto a los valores del enfoque anterior, que de por sí ya se habían alcanzado métricas realmente satisfactorias. En este caso, el valor de las funciones de pérdida y accuracy para el dataset de validación fue muy similar en comparación con el enfoque anterior, aunque con una leve ventaja para este enfoque. La gran diferencia observada al finalizar el entrenamiento del modelo fue una fuerte ventaja en las métricas para el dataset de entrenamiento. En la Figura 5.9 se puede apreciar que los valores para la pérdida se mantuvieron siempre por debajo de 0.03, una vez que el proceso de optimización comenzó a converger. Para la precisión los valores estuvieron también permanentemente en torno a 0.99, lo cual son valores sorprendentes. Este tipo de métricas puede inferir que existió overfitting del modelo, pero al observar las funciones asociadas al dataset de validación, se puede comprobar que el mismo es inexistente. Generalmente una alarma para determinar la existencia de overfitting es la función de costo para la validación. Al iniciar el entrenamiento suele descender rápidamente, desacelerando en el descenso conforme avanzan las pasadas. En un punto lo que sucede es que la curva de costo comience a crecer, no en casos aislados, sino que crezca de forma sostenida. Este tipo de situación es un indicador muy robusto de existencia de overfitting. Puede suceder que las características del dataset de validación provoquen este fenómeno, pero en ese caso el problema no sería del modelo sino del dataset, ya que tanto el dataset de entrenamiento de validación deben provenir de la misma distribución de información para poder obtener métricas fiables.

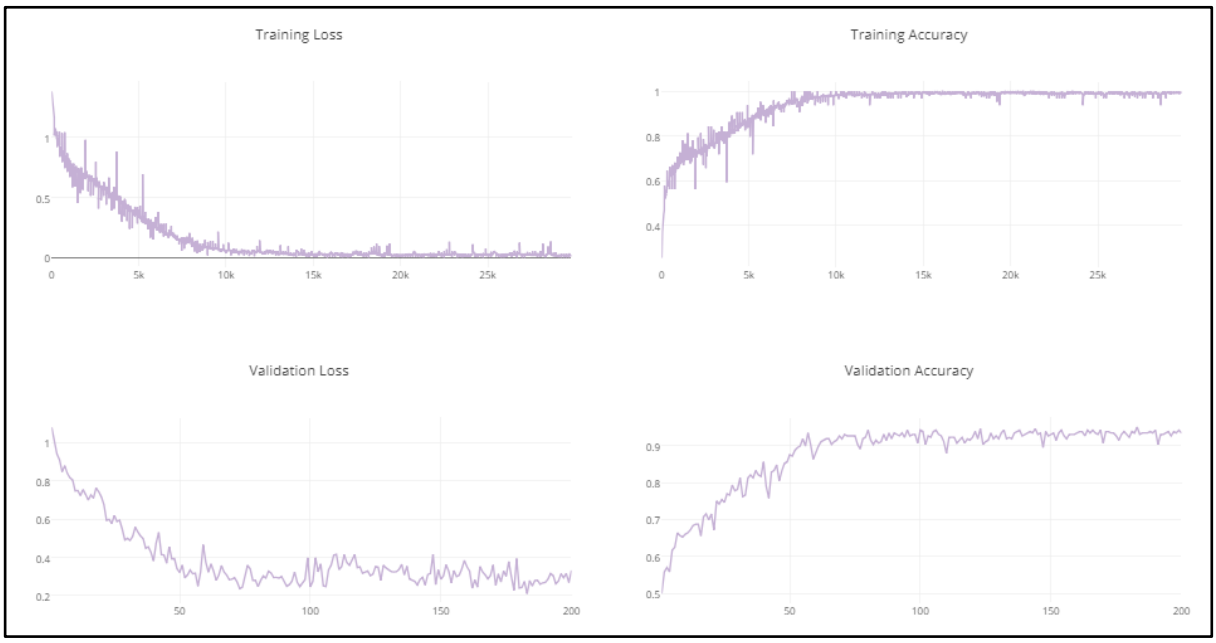

Figura 5.9. Métricas resultantes del entrenamiento del enfoque de redes neuronales convolucionales recurrentes.

5.5. Conclusiones

A modo de resumen sobre las arquitecturas desarrolladas para la resolución del problema de predecir una categoría dada, se provee la Figura 5.10 en donde se efectúa una comparación entre las métricas obtenidas para el entrenamiento de cada modelo. En la Tabla 5.5 se muestra la asociación de los modelos con la identificación que la gráfica le otorga a cada enfoque.

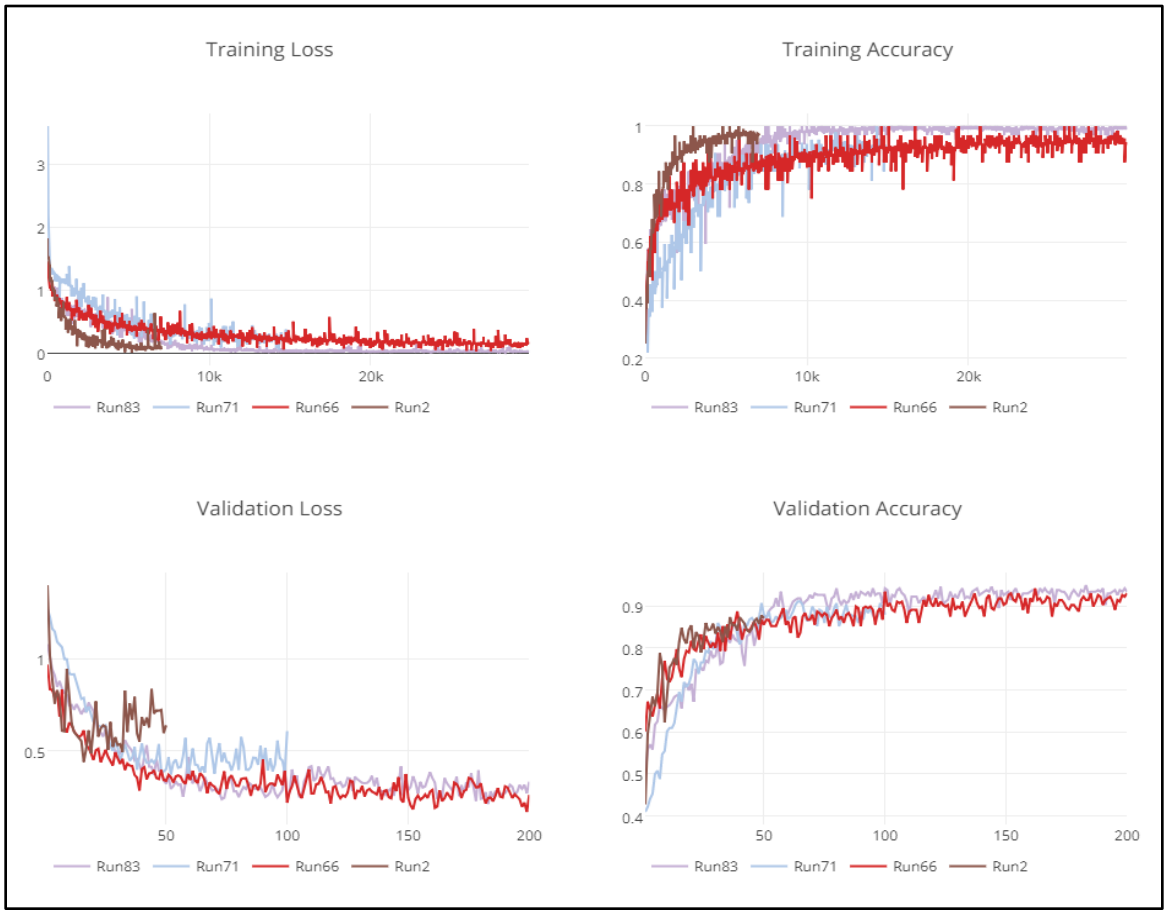

Figura 5.10. Comparación de las métricas de los cuatro enfoques

Enfoque	Identificación en la Gráfica
Capas Densas	Run 2
Capas Convolucionales + Densas	Run 71
Capas Recurrentes + Densas	Run 66
Capas Conv. + Recurrentes + Densas	Run 83

Tabla 5.5. Asociación entre la identificación de la Figura 5.10 con cada enfoque empleado.

A continuación, se provee una tabla en donde se facilita el análisis sobre las métricas para cada enfoque. Se podrá observar que algunos enfoques resultaron ser mejores en algunos aspectos, pero peores en otros. Partiendo de esa idea, se le provee al usuario final la posibilidad de elegir el enfoque con el cual quiera efectuar la clasificación de los mensajes originales. Habiendo observado la comparación entre las métricas de cada enfoque, se puede observar que en la Tabla 5.6 para tres de cuatro funciones, el último enfoque que emplea capas convolucionales y recurrentes es el mejor de los cuatro.

Función	Mejor Enfoque
Training Loss	Enfoque Convolucionales + Recurrentes
Training Accuracy	Enfoque Convolucionales + Recurrentes
Validation Loss	Enfoque Recurrentes
Validation Accuracy	Enfoque Convolucionales + Recurrentes

Tabla 5.6. Análisis del mejor enfoque para cada función.

Capítulo 6: Experimentación

En este capítulo se describe la experimentación desarrollada para evaluar los modelos propuestos. En la primera sección se hace referencia a la configuración de la experimentación. Es decir que se proveen detalles inherentes al conjunto de datos que se utiliza como objeto de prueba. Se hace mención a diversos aspectos del mismo como por ejemplo la cantidad de instancias con la que se cuenta, distribución de las categorías, contexto de donde proviene el dataset, etc. Junto con la descripción del conjunto de datos, se tocan otros temas como por ejemplo las limitaciones del mismo, sobre todo aquellas dificultades que, en caso de no presentarse, podrían haber permitido mejores etapas de entrenamiento en el modelo predictivo.

En la Sección 6.2. se abordan todos los aspectos relacionados a los resultados obtenidos, a partir del dataset empleado en la sección anterior. Se proveen diversas métricas asociadas a los resultados de la experimentación, estando entre ellas las matrices de confusión, junto con el correspondiente conjunto de métricas que se desprende de dicha matriz. A partir de las métricas que se presentan, se efectúa un análisis general concluyendo cuán satisfactorio fue el resultado obtenido. Para ello se evalúa cada uno de los cuatro enfoques de redes neuronales implementados en el capítulo anterior.

Luego se realiza un análisis más abarcativo de los resultados obtenidos previamente. Se hace uso de la herramienta desarrollada, en la cual se muestra el armado de los perfiles de usuario a partir de los mensajes provistos. A partir de los resultados obtenidos se detalla la forma en que los gráficos que se muestran, asistiendo a un hipotético supervisor en el armado de grupos de trabajo, detección de grupos conflictivos, etc.

Por último, en la Sección 6.4, se efectúa un resumen de todas las cuestiones vistas durante este capítulo, resaltando aquellos aspectos más relevantes de la experimentación y de los resultados. A partir de los resultados obtenidos, se construye una conclusión del tema desarrollado, a fin de pasar en limpio el tema investigado.

6.1. Configuración

En esta sección se abarcan todos los detalles relacionados a los conjuntos de datos empleados en este trabajo de grado. Como bien se indicó en el Capítulo 4 con la proposición de la solución, el objetivo es el armado de perfiles de usuario a partir de un conjunto de mensajes provistos por un usuario. A partir de dichos mensajes, se obtiene una categorización de cada uno de ellos, que en este caso constituye una conducta IPA. Esto quiere decir que, el modelo predictivo tiene como entrada un determinado mensaje, mientras que como salida tiene una conducta. Al momento de entrenar las redes neuronales que realicen esta categorización, se empleó un dataset con este formato. Es decir, que el dataset utilizado consistió de un listado de mensajes, estando cada uno de ellos etiquetados con su correspondiente conducta IPA, que va del 1 al 12. El origen de los mismos proviene de un trabajo de grado realizado por Martín Mineo [47] en donde, entre las tareas que componen el trabajo que realizó, se encuentra un listado de casi 2500 mensajes etiquetados, provenientes de diversos grupos de trabajo.

Una de las características salientes de las técnicas de aprendizaje profundo, es la cantidad de datos que necesita. Una red neuronal por más afinada que tenga su arquitectura e hiperparámetros, no podrá ajustarse correctamente a los datos si la cantidad de los mismos no es la adecuada. En caso de no lograrse, las métricas de precisión y costo no serán lo suficientemente óptimas como para poner el modelo en producción, debido a la existencia de overfitting o underfitting. Para paliar esta cuestión, es decir para mejorar las métricas que el entrenamiento de la red arrojaba, se recurrió a diversas técnicas de preprocesamiento del dataset, en donde se eliminaron mensajes duplicados, errores de ortografía, entre otras cosas. No obstante, las métricas no mejoraron en demasía, por lo que la cuestión original se hizo más notoria: se necesitaban más datos.

A partir de esta problemática se recurrió a lo que se conoce como *data augmentation*[14], que consiste en realizar operaciones sobre los datos existentes a fin de producir más ejemplos. Un ejemplo de esto es tomar por ejemplo una imagen, e invertirla, para así contar con más ejemplos disponibles. Se trata de un procedimiento complejo, que debe realizarse cuidadosamente para que se puedan obtener resultados positivos. En casos de no realizarse correctamente, las métricas que se pueden obtener pueden ser peores debido a la reducción de la capacidad de aprendizaje por causa de la repetición de los datos de entrada.

En este caso, al trabajar con texto, se optó por duplicar la cantidad de ejemplos mediante la técnica que se describe a continuación. Se tomó cada mensaje, y se lo tradujo a otro idioma utilizando un servicio especializado en esa tarea. Una vez que se tiene el dataset traducido, se lo vuelve a traducir al español. El detalle en cuestión es que, en los servicios de traducción, el proceso no es simétrico. Esto quiere decir que en caso de tener un mensaje A, y se lo traduce a su correspondiente B en otro idioma, al querer traducirlo nuevamente al idioma original, es muy probable que la traducción sea equivalente, pero no igual. De esta manera, se tiene el dataset duplicado a aproximadamente 5000 mensajes, en donde si bien hay mensajes con el mismo significado, textualmente son diferentes. Esta técnica resultó ser muy útil y determinante al momento de entrenar las redes neuronales, repercutiendo en una capacidad de aprendizaje sumamente mejorada, en comparación con la versión original del dataset.

Otro aspecto a destacar de los dataset es la distribución de las conductas. En un grupo de trabajo, los integrantes que componen al mismo tienen una determinada tarea en común, y se supone que todos interactúan de forma tal que se garantice el cumplimiento la tarea en cuestión. Este tipo de contexto provoca que el dataset con el que se cuenta, esté desbalanceado en cuanto a la cantidad de mensajes etiquetados de cada conducta. Por ejemplo, la cantidad de mensajes etiquetados de conductas negativas, es mucho menor que la cantidad de mensajes de conductas positivas o de respuesta. Esto provocó que se haya tenido que efectuar un leve balanceo de clases, eliminando ejemplos redundantes que pertenezcan a categorías con ejemplos excesivos. Esta decisión también contribuyó al entrenamiento de las redes, permitiendo que aprendan mejor a predecir aquellas categorías con menor cantidad de ejemplos.

[14] https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced

A continuación, se proveen tablas para indicar el balanceo que se realizó sobre el dataset ya aumentado, es decir ya habiendo sido sometido al servicio de traducción. En dichas tablas se muestran las distribuciones de categorizaciones para el dataset sin balancear, y balanceado, respectivamente. Además de las distribuciones para cada conducta, se muestra una tabla más, agrupando las conductas por reacciones, a fin de facilitar la visualización del desbalance de clases.

Como se podrá observar en las tablas 6.1 y 6.2, si bien para algunas categorías se eliminaron mensajes redundantes, en otras se duplicaron o triplicaron los mensajes. Esta técnica si bien puede ser peligrosa, fue necesaria para obtener un dataset más balanceado. El overfitting que puede producir la duplicación de estas clases se solucionó mediante una regularización (léase Dropout rate, etc.) más fuerte al momento de desarrollar las arquitecturas de las redes neuronales.

A modo de evitar comparar permanentemente las primeras dos tablas con las dos últimas, se agrega la cantidad original de mensajes seguido a la cantidad nueva, en las Tablas 6.1 y 6.2. Para el caso de los mensajes etiquetados con la conducta C1 y C2, no se realizó balanceo alguno.

Conducta IPA	**Cantidad de Mensajes**
C1. Muestra Solidaridad	540
C2. Muestra Relajamiento	194
C3. Muestra Acuerdo	436 (antes 636)
C4. Da Sugerencia	540 (antes 640)
C5. Da Opinión	534 (antes 634)
C6. Da Información	576 (antes 1076)
C7. Pide Información	548 (antes 274)
C8. Pide Opinión	500 (antes 250)
C9. Pide Sugerencia	144 (antes 72)
C10. Muestra Desacuerdo	402 (antes 134)
C11. Muestra Tensión	498 (antes 166)
C12. Muestra Antagonismo	78 (antes 26)

Tabla 6.1. Distribución de conductas del dataset habiendo aplicado balanceo de clases.

Reacción	Cantidad de Mensajes	Proporción ACTUAL	Proporción Original (Antes)
Positiva	1170 (antes 1370)	0.234	0.295
Responde	1650 (antes 2350)	0.331	0.506
Pregunta	1192 (antes 596)	0.239	0.128
Negativa	978 (antes 326)	0.196	0.071

Tabla 6.2. Distribución de reacciones del dataset habiendo aplicado balanceo de clases. Se trata de una agrupación de las conductas por reacciones, para facilitar la visualización.

En esta última tabla puede apreciarse claramente el balanceo de clases obtenido con respecto a la distribución original. La tercer columna se utiliza para comparar las proporciones de cada clase, en este caso se utilizó la tabla con las reacciones debido a que facilita más la comprensión de la técnica aplicada.

Por último, se detallan las métricas a emplear en la siguiente sección, las cuales consisten en: accuracy, recall, precision, y F1 Score.

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

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

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

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN}$$

A modo de clarificación, se tienen las siguientes referencias:

- TP: True Positives (Verdaderos Positivos)
- TN: True Negatives (Verdaderos Negativos)
- FP: False Positives (Falsos Positivos)
- FN: False Negatives (Falsos Negativos)
- P: Positives (Suma de los positivos)
- N: Negatives (Suma de los negativos)

6.2. Resultados con Enfoques de Aprendizaje

A partir del dataset anterior, se entrenaron las redes neuronales que predicen una conducta asociada a cada mensaje. Sin embargo, la finalidad del trabajo de grado no son las conductas IPA en sí, sino los indicadores Symlog. A partir de los mapeos propuestos en el Capítulo 3, se arman estos indicadores tomándose en ambos casos, las conductas IPA predichas por los modelos predictivos.

Con respecto a los indicadores Symlog, se procedió de la siguiente manera. Para el armado de los mismos, se tiene como entrada doce valores, es decir, el recuento de las doce conductas predichas por la red neuronal, para un determinado usuario en una determinada sesión de trabajo. Dichas conductas se agrupan en reacciones, y a partir de dichas reacciones se arman los seis indicadores Symlog. Como bien indica la teoría Symlog, se tienen tres dimensiones que discretizan la personalidad de una persona. A partir de los seis indicadores, se generaron las tres dimensiones, ya que cada dimensión está compuesta por dos indicadores. Puesto a modo de ejemplo, los dos primeros indicadores (Dominante y Sumiso) definen a la dimensión U/D, por lo que dicha dimensión se calculó mediante la resta del segundo indicador sobre el primero. Para el resto de las dimensiones se procede de forma análoga con los indicadores correspondientes.

Una vez que se tienen las dimensiones Symlog, se las compara con las dimensiones Symlog que figuran en una serie de encuestas completadas por diversas personas. Estas encuestas consisten en una serie de preguntas a responder, en donde cada pregunta tiene una cierta influencia sobre una determinada dimensión. Estas encuestas fueron discretizadas a modo de poder comparar automáticamente los valores que arrojan las mismas, contra los mapeos que se realizan en este trabajo de grado, a partir de la predicción de las conductas IPA.

Las pruebas que se realizaron se llevaron a cabo de la siguiente manera. No se compara el valor final para cada dimensión contra los obtenidos, debido a un problema de escala. La ponderación que se le da a la respuesta de cada pregunta en las encuestas, no es la misma que la que se usa en la solución propuesta, que trabaja a base de la contabilidad de conductas IPA. La decisión empleada para realizar la comparación fue comparar los resultados obtenidos dimensión a dimensión. Esto quiere decir que se obtendrá una serie de métricas asociada a cada una de las tres dimensiones. Como bien indica la teoría, cuando el valor de una determinada dimensión sale de un intervalo, indica una intensidad relacionada a una dimensión. Por ejemplo, para la dimensión U/D, en caso de tener un valor de 5, está por fuera del intervalo [3, -3], por lo tanto, en esa dimensión, se señala una

U, y en caso de tener, por ejemplo -4, se señala una D. En casos donde el valor cae dentro del intervalo, no se señala con ninguna letra.

A modo de clarificación, las comparaciones de similitud entre la predicción y el valor de la encuesta se realiza de la siguiente manera. Si una persona tiene las dimensiones IPA "UP", quiere decir que sus indicadores se excedieron por encima del intervalo en las dimensiones U/D y P/N, mientras que para la dimensión F/B, el valor se mantuvo contenido en el intervalo [3, -3]. De esta manera, la comparación de acierto se realiza a nivel dimensión. Si en la predicción se tiene por ejemplo un perfil "UPF", pero en la encuesta esa misma persona posee "UF", quiere decir que el acierto se dio en la dimensión U/D y F/B. A modo de aclaración, si para una determinada dimensión, la predicción no arroja ninguna letra (es decir que se mantuvo dentro del intervalo), y en la encuesta tampoco arroja una letra en la misma dimensión, también se considera un acierto. Por ejemplo, si se tiene "UP" en la predicción, y "UN" en la encuesta, quiere decir que en la dimensión F/B también hubo un acierto.

Dicho esto, para cada dimensión se construye una matriz de confusión de 3x3, ya que, para cada dimensión, hay tres valores posibles. En el caso de la dimensión U/D, se tienen "U", "-" y "D". En total se tienen tres matrices, las cuales se muestran a continuación en las Tablas 6.3, 6.4 y 6.5. Todas las tablas que se muestran hacen referencia al enfoque más básico de los cuatro propuestos, es decir el que utiliza solamente redes densas. La idea es mostrar la variación de las métricas obtenidas en función de que varía la complejidad de la arquitectura del enfoque.

6.2.1. Tablas correspondientes al enfoque basado en Redes Densas

(U/D) →	U	-	D
U	5	15	0
-	58	43	0
D	0	6	0

Tabla 6.3. Matriz de confusión para la dimensión U/D (up/down), que involucra los indicadores de dominancia y sumisión. La flecha indica valores predichos.

(P/N) →	P	-	N
P	89	9	0
-	26	2	0
N	1	0	0

Tabla 6.4. Matriz de confusión para la dimensión P/N (positive/negative), que involucra los indicadores de amistoso y no amistoso. La flecha indica valores predichos.

(F/B) →	F	-	B
F	48	9	0
-	58	12	0
B	0	0	0

Tabla 6.5. Matriz de confusión para la dimensión F/B (forward/backward), que involucra los indicadores de dominancia y sumisión. La flecha indica valores predichos.

Como se podrá observar, la cantidad de ejemplos en cada matriz es la misma para las tres. Esto quiere decir que, para las conductas de cada persona, se descompuso el mapeo de Symlog en tres dimensiones, es decir que cada persona tiene tres valores de Symlog, correspondiente a cada una de las tres dimensiones.

Sin embargo, la dimensión de este tipo de matrices no permite que se puedan calcular las métricas correspondientes de forma directa. Para poder calcular diversas métricas como por ejemplo accuracy o recall, es necesario que las matrices de confusión sean binarias, por lo tanto, se deben descomponer estas tres matrices en tres matrices cada una. Por ejemplo, para la dimensión U/D, se obtienen tres matrices de confusión, en donde cada una indica positivo o negativo de “U”, “-”, o “D”.

A modo de evitar la sobrecarga de información en la Sección, las matrices binarias, que constituyen ser tres para cada dimensión, para donde se tienen cuatro enfoques, no se incluyen en el presente capítulo, sino que se incluyen en el Anexo I. En dicho anexo se puede encontrar para cada enfoque, las matrices de confusión binarias para cada dimensión, que justifican las medidas expuestas en esta sección.

A partir de las tablas recientemente expuestas (y de las anexadas), se permite el cálculo de las métricas asociadas a la predicción/mapeo de la dimensión U/D. Las métricas empleadas fueron las siguientes: accuracy, recall, precision, F1 Score, las cuales fueron definidas sobre el final de la sección anterior. Si bien para cada matriz de confusión se calcularon las cuatro métricas, la que más relevancia tiene, y la que más se tuvo en cuenta fue el accuracy debido a que el resto de las medidas, al trabajar solo con los verdaderos positivos (true positives) en el numerador, provoca que en las matrices que no haya dicho tipo de valores, el resultado sea cero. Esto no se da por falencias en el modelo predictivo o en el mapeo de las conductas, sino debido a inexistencia de encuestas que indiquen, por ejemplo, una N en el indicador final.

En la Tabla 6.6 se muestran las métricas obtenidas para cada una de las matrices de confusión binarias que se desprenden de las anteriormente expuestas.

Tablas (U/D)	Accuracy	Recall	Precisión	F1 Score
(U/ No U)	0.425	0.250	0.079	0.135
(-/ No -)	0.378	0.426	0.672	0.577
(D/ No D)	0.953	0	s/c[15]	0

Tabla 6.6. Métricas obtenidas a partir de las matrices de confusión asociadas a la dimensión U/D, relacionada con los indicadores de dominancia y sumisión.

Como se puede observar, existen algunas métricas con valores muy bajos o bien nulos, que se debe a la inexistencia de valores positivos para el indicador en cuestión. Estas situaciones le restan relevancia a la métrica debido a que, para realizar las experimentaciones en conjunto, tanto para esta dimensión como para el conjunto de las tres dimensiones, se realiza un promedio de cada métrica. En casos donde el dataset no provee casos de una determinada dimensión, se provoca que los valores obtenidos en las diversas matrices de confusión perjudiquen seriamente el promedio. Al momento de la experimentación, la métrica que más relevancia tuvo, y la que más confianza otorga es la de accuracy, por lo tanto, debiera ser la más importante para observar.

A continuación, se procede de forma análoga para el resto de las dimensiones, comenzado por la P/N para luego proceder para la F/B. Se muestran las métricas obtenidas para cada matriz binaria desprendida, para luego proceder a efectuar el promedio resultante.

[15] s/c: Sin Calcular debido a que el valor del denominador no permite el cálculo de la métrica.

Tablas (P/N)	Accuracy	Recall	Precision	F1 Score
(P/No P)	0.717	0.908	0.767	0.942
(-/No -)	0.724	0.071	0.182	0.033
(N/No N)	0.992	0	s/c	0

Tabla 6.7. Métricas obtenidas a partir de las matrices de confusión asociadas a la dimensión P/N, relacionada con los indicadores de "amistoso" y "no amistoso".

Tablas (F/B)	Accuracy	Recall	Precision	F1 Score
(F/No F)	0.472	0.842	0.453	0.821
(-/No -)	0.472	0.171	0.571	0.185
(B/No B)	1	s/c	s/c	0

Tabla 6.8. Métricas obtenidas a partir de las matrices de confusión asociadas a la dimensión P/N, relacionada con los indicadores de "amistoso" y "no amistoso".

Dadas las cuestiones presentadas con respecto a la falta de valores positivos, algunas métricas además de dar cero, no pueden ser calculadas debido a que también dan cero en el denominador. Esto provocó que, al momento de calcular el promedio de todas ellas, el resultado obtenido además de ser bajo, no resultó ser del todo confiable debido a las características del dataset con el que se trabajó. La métrica final con la que se eligió trabajar es la de *accuracy*, la cual obteniendo el accuracy asociado a la predicción de cada una de las tres dimensiones, se pudo calcular un promedio, el cual se detalla a partir de los siguientes datos:

Accuracy U/D	Accuracy P/N	Accuracy F/B	Accuracy FINAL
0.585	0.811	0.648	0.682

Tabla 6.9. Valores de accuracy para cada dimensión, junto con el correspondiente promedio.

6.2.2. Tablas correspondientes al enfoque basado en Redes Convolucionales

En este enfoque se obtuvieron resultados mucho mejores en comparación con el enfoque anterior, obteniendo un accuracy final superior al del enfoque de redes con capas densas. La obtención del mismo se da a partir de las matrices mostradas a continuación:

(U/D) →	U	-	D
U	3	17	0
-	18	81	2
D	0	6	0

Tabla 6.10. Matriz de confusión para la dimensión U/D (up/down), que involucra los indicadores de dominancia y sumisión. La flecha indica valores predichos.

Tablas (U/D)	Accuracy	Recall	Precision	F1 Score
(U/ No U)	0.724	0.150	0.143	0.054
(-/ No -)	0.661	0.802	0.779	0.876
(D/ No D)	0.937	0	0	0

Tabla 6.11. Métricas obtenidas a partir de las matrices de confusión asociadas a la dimensión U/D, relacionada con los indicadores de dominancia y sumisión.

(P/N) →	P	-	N
P	96	2	0
-	27	1	0
N	1	0	0

Tabla 6.12. Matriz de confusión para la dimensión P/N (positive/negative), que involucra los indicadores de amistoso y no amistoso. La flecha indica valores predichos.

Tablas (P/N)	Accuracy	Recall	Precision	F1 Score
(P/No P)	0.764	0.980	0.774	0.985
(-/No -)	0.772	0.036	0.333	0.016
(N/No N)	0.992	0	s/c	0

Tabla 6.13. Métricas obtenidas a partir de las matrices de confusión asociadas a la dimensión P/N, relacionada con los indicadores de "amistoso" y "no amistoso".

(F/B) →	F	-	B
F	41	16	0
-	46	24	0
B	0	0	0

Tabla 6.14. Matriz de confusión para la dimensión F/B (forward/backward), que involucra los indicadores de dominancia y sumisión. La flecha indica valores predichos.

Tablas (F/B)	Accuracy	Recall	Precision	F1 Score
(F/No F)	0.512	0.719	0.471	0.672
(-/No -)	0.512	0.343	0.600	0.356
(B/No B)	1	s/c	s/c	0

Tabla 6.15. Métricas obtenidas a partir de las matrices de confusión asociadas a la dimensión F/B, relacionada con los indicadores de "amistoso" y "no amistoso".

Accuracy U/D	Accuracy P/N	Accuracy F/B	Accuracy FINAL
0.774	0.843	0.675	0.764

Tabla 6.16. Valores de accuracy para cada dimensión, junto con el correspondiente promedio.

6.2.3. Tablas correspondientes al enfoque basado en Redes Recurrentes

En este enfoque si bien se obtuvieron resultados satisfactorios en cuanto a accuracy, llamativamente las métricas se encontraron sensiblemente por debajo de las obtenidas en el enfoque de redes convolucionales. Esto se da de forma contraria al entrenamiento de las redes, en donde las funciones de precisión y pérdida tanto para el entrenamiento como para la validación habían arrojado mejores resultados en el enfoque de recurrentes, en comparación con su contrapartida de convolucionales.

(U/D) →	U	-	D
U	6	14	0
-	35	65	1
D	0	6	0

Tabla 6.17. Matriz de confusión para la dimensión U/D (up/down), que involucra los indicadores de dominancia y sumisión. La flecha indica valores predichos.

Tablas (U/D)	Accuracy	Recall	Precision	F1 Score
(U/ No U)	0.614	0.300	0.146	0.122
(-/ No -)	0.559	0.644	0.765	0.756
(D/ No D)	0.945	0	0	0

Tabla 6.18. Métricas obtenidas a partir de las matrices de confusión asociadas a la dimensión U/D, relacionada con los indicadores de dominancia y sumisión.

(P/N) →	P	-	N
P	95	3	0
-	27	1	0
N	1	0	0

Tabla 6.19. Matriz de confusión para la dimensión P/N (positive/negative), que involucra los indicadores de amistoso y no amistoso. La flecha indica valores predichos.

Tablas (P/N)	Accuracy	Recall	Precision	F1 Score
(P/No P)	0.756	0.969	0.772	0.979
(-/No -)	0.764	0.036	0.250	0.016
(N/No N)	0.992	0	s/c	0

Tabla 6.20. Métricas obtenidas a partir de las matrices de confusión asociadas a la dimensión P/N, relacionada con los indicadores de "amistoso" y "no amistoso".

(F/B) →	F	-	B
F	44	13	0
-	50	20	0
B	0	0	0

Tabla 6.21. Matriz de confusión para la dimensión F/B (forward/backward), que involucra los indicadores de "orientación a la tarea" y "orientación a lo socioemocional". La flecha indica valores predichos.

Tablas (F/B)	Accuracy	Recall	Precision	F1 Score
(F/No F)	0.504	0.772	0.468	0.727
(-/No -)	0.504	0.286	0.606	0.299
(B/No B)	1	s/c	s/c	0

Tabla 6.22. Métricas obtenidas a partir de las matrices de confusión asociadas a la dimensión F/B, relacionada con los indicadores de "orientación a la tarea" y "orientación a lo socioemocional".

Accuracy U/D	Accuracy P/N	Accuracy F/B	Accuracy FINAL
0.706	0.992	0.669	0.738

Tabla 6.23. Valores de accuracy para cada dimensión, junto con el correspondiente promedio.

6.2.4. Tablas correspondientes al enfoque basado en Redes Convolucionales Recurrentes

Este último enfoque constituye el mejor de los cuatro al mostrarse superior a los demás, no solo en cuanto a las métricas correspondientes al entrenamiento de la red neuronal, sino también en el accuracy resultante, siendo superior al mejor enfoque hasta el momento, el de convolucionales.

(U/D) →	U	-	D
U	2	18	0
-	6	93	2
D	0	6	0

Tabla 6.24. Matriz de confusión para la dimensión U/D (up/down), que involucra los indicadores de dominancia y sumisión. La flecha indica valores predichos.

Tablas (U/D)	Accuracy	Recall	Precision	F1 Score
(U/ No U)	0.811	0.100	0.250	0.033
(-/ No -)	0.748	0.921	0.795	.0949
(D/ No D)	0.937	0	0	0

Tabla 6.25. Métricas obtenidas a partir de las matrices de confusión asociadas a la dimensión U/D, relacionada con los indicadores de dominancia y sumisión.

(P/N) →	P	-	N
P	98	0	0
-	26	2	0
N	1	0	0

Tabla 6.26. Matriz de confusión para la dimensión P/N (positive/negative), que involucra los indicadores de amistoso y no amistoso. La flecha indica valores predichos.

Tablas (P/N)	Accuracy	Recall	Precision	F1 Score
(P/No P)	0.787	1	0.784	0.990
(-/No -)	0.795	0.071	1	0.031
(N/No N)	0.992	0	s/c	0

Tabla 6.27. Métricas obtenidas a partir de las matrices de confusión asociadas a la dimensión P/N, relacionada con los indicadores de "amistoso" y "no amistoso".

(F/B) →	F	-	B
F	45	12	0
-	41	29	0
B	0	0	0

Tabla 6.28. Matriz de confusión para la dimensión F/B (forward/backward), que involucra los indicadores de "orientación a la tarea" y "orientación a lo socioemocional". La flecha indica valores predichos.

Tablas (F/B)	Accuracy	Recall	Precisión	F1 Score
(F/No F)	0.583	0.789	0.523	0.687
(-/No -)	0.583	0.414	0.707	0.403
(B/No B)	1	s/c	s/c	0

Tabla 6.29. Métricas obtenidas a partir de las matrices de confusión asociadas a la dimensión F/B, relacionada con los indicadores de "orientación a la tarea" y "orientación a lo socioemocional".

Accuracy U/D	Accuracy P/N	Accuracy F/B	Accuracy FINAL
0.832	0.858	0.722	**0.804**

Tabla 6.30. Valores de accuracy para cada dimensión, junto con el correspondiente promedio.

Como se puede observar en esta última tabla, el accuracy obtenido a través del enfoque híbrido de convolucionales y recurrentes, obtiene los mejores resultados no solo para la clasificación de las conductas, sino también para el armado de los perfiles, a través del mapeo propuesto. De esta manera, al momento de que el usuario final emplee la aplicación web desarrollada, se recomendará que utilice este último enfoque. No obstante, igualmente se le proporcionará la posibilidad de que pueda generar los perfiles utilizando cualquier de los enfoques propuestos, sea por preferencia o para posibilitar la realización de diversos análisis comparativos entre los enfoques propuestos.

6.3. Análisis de Métricas de Usuarios

Hasta el momento se analizaron los resultados obtenidos por el framework, pero a nivel crudo, es decir a nivel puramente de instancia. Si bien se mostraron todos los resultados relacionados con la experimentación, la misma no se realizó a través de la aplicación web desarrollada. En esta sección se realizarán experimentación de aplicación, utilizando como ejemplo un dataset de mensajes en una de las fuentes soportadas, sometiéndolo a los diversos enfoques para así obtener los correspondientes perfiles de usuario. Como la finalidad del trabajo de grado es que la implementación pueda ser aprovechada por usuarios finales que no deben provenir necesariamente del área de sistemas, es de especial importancia que la información se muestre de una forma amigable con el usuario.

El dataset empleado para la experimentación consiste en tres sesiones de trabajo, correspondientes a grupos de Facebook [38]. En dichos grupos de colaboración, los integrantes del mismo poseen un interés en común, por lo tanto, se lo puede considerar como un grupo de trabajo. Cada sesión de trabajo está representada en formato ARFF, en donde en total se tienen casi 3000 mensajes. Dada la naturaleza de los grupos de Facebook, en donde la organización de la misma está dada por un listado de posts y comentarios al mismo, cada posteo y/o comentario es considerado como un mensaje. Por ejemplo, si un determinado grupo posee solamente un post con nueve comentarios, el dataset contiene diez mensajes, en donde el primero se corresponde con el contenido del post, mientras que los nueve siguientes se corresponden con los comentarios del post en cuestión. De esta manera el formato de representación de la información de los grupos de Facebook se unifica a fin de compatibilizarlo con el framework para generar perfiles de usuario asociados a cada usuario del grupo de trabajo. En este caso, perfiles de usuario se asocian a cada perfil de Facebook.

Como se puede observar, el origen de la información de los mensajes puede ser indistinto, lo que se solicita es emplear cualquiera de los formatos soportados por el framework para poder generar los perfiles de usuario. En el dataset ejemplificado se optó por usar ARFF, pero la información puede venir también en forma de Takeout, CSV, de una base de datos, etc. Lo que se necesita en cualquiera de los casos es, en caso de disponer de la información en otro formato, efectuar rutinas de conversión de formato. Para estos casos se recomienda siempre hacer la conversión a CSV debido a razones de performance, ya que se trata del formato nativo que emplea el servicio de las redes neuronales, por lo tanto, en ese caso no se deben realizar pasos adicionales para llevar a cabo la conversión de formato a través del servicio correspondiente.

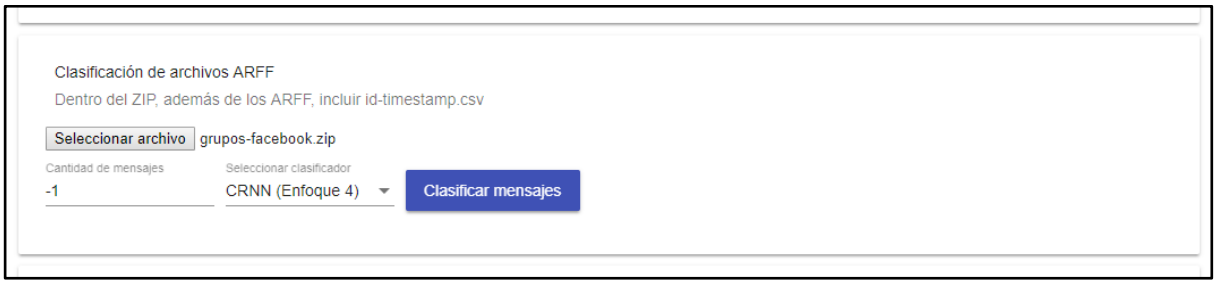

Figura 6.1. Tarjeta en donde se provee el dataset con formato ARFF para su posterior clasificación.

En la Figura 6.1 se muestra la porción de la aplicación web en donde se efectúa la carga de información, siendo para este caso la carga de ARFF. Como se puede observar, se trabaja con un archivo ZIP, que contiene un ARFF correspondiente a cada una de las sesiones de trabajo. Además, se debe proveer, también dentro del ZIP, un CSV que simplemente se encargue de determinar la marca de tiempo asociada a cada sesión. Luego además de la cantidad de mensajes a clasificar ("-1" indica todos los mensajes del dataset provisto), se pide clasificar los mensajes con el enfoque que mejor resultados ha dado: el de convolucionales con recurrentes (CRNN). Al presionar el correspondiente botón, se envía el contenido al backend para que genere los perfiles. En las sucesivas figuras que se exponen a continuación, se puede observar el resultado del proceso de clasificación, que consiste en el perfil asociado a cada usuario presente en el dataset. Los indicadores se muestran en forma de tabla, en donde cada fila se corresponde con un usuario, mientras que cada tipo de indicador se ubica en una columna. También se muestran los gráficos que separan los indicadores que se muestran en las tablas, en indicadores por sesión de trabajo, estando las mismas ordenadas por la marca de tiempo asociada a cada sesión. Primero se exponen los gráficos con los indicadores separados, y luego para los mismos ejemplos, se muestran las tablas que agrupan los indicadores que cada usuario obtuvo en cada sesión de trabajo.

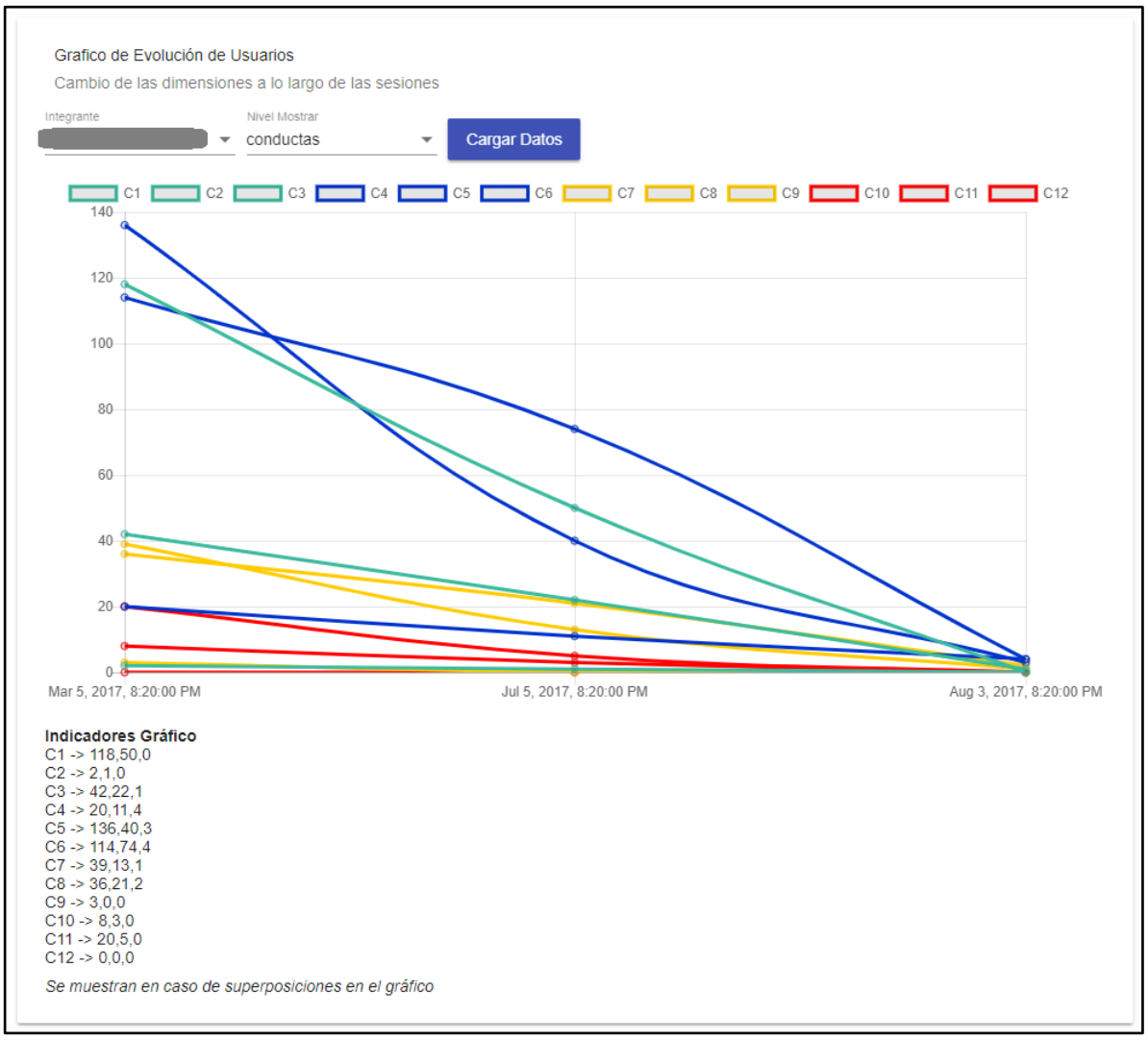

Figura 6.2. Gráfico con las conductas predichas por la red neuronal en su mejor enfoque (CRNN).

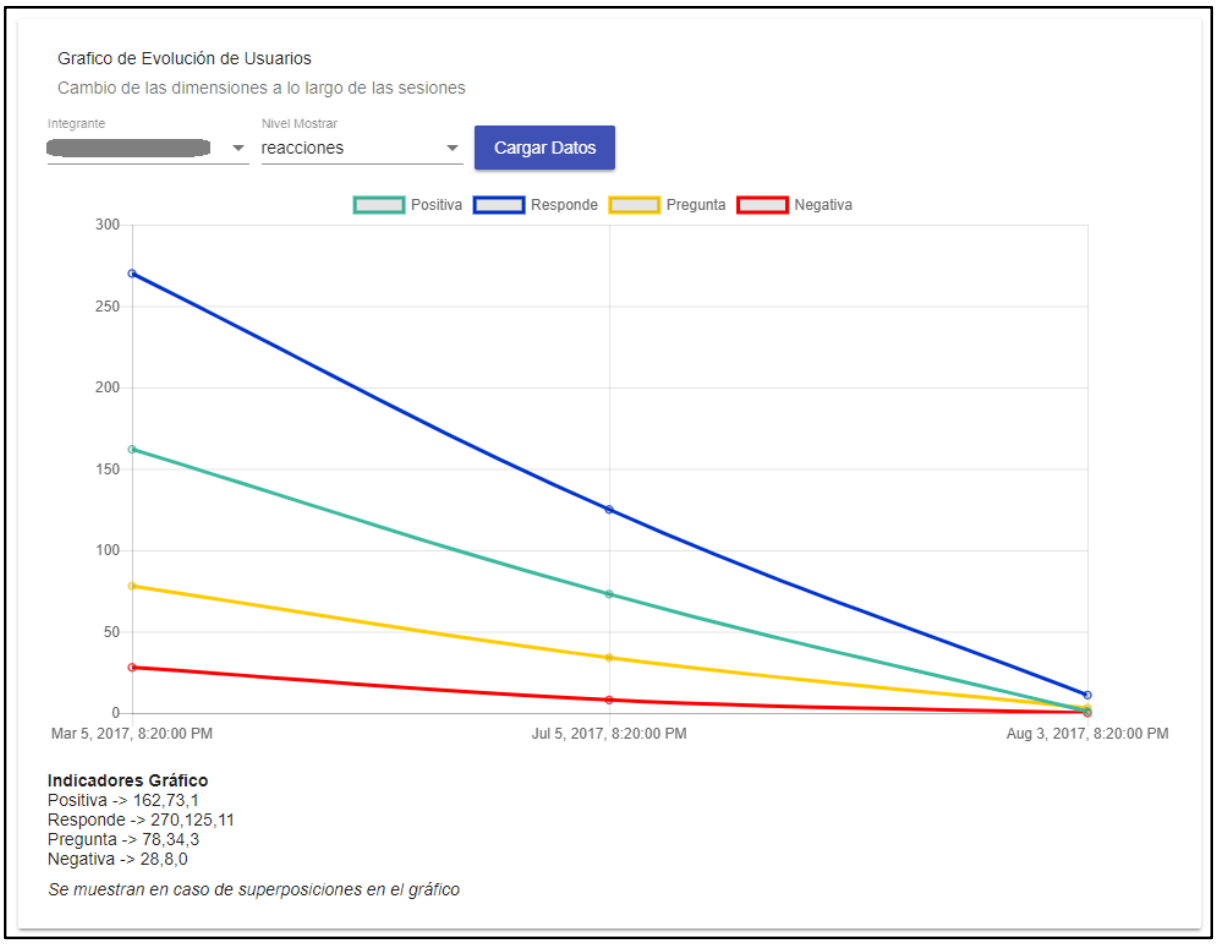

Figura 6.3. Gráfico con las reacciones resultantes de agrupar las conductas. Los colores de las mismas se corresponden.

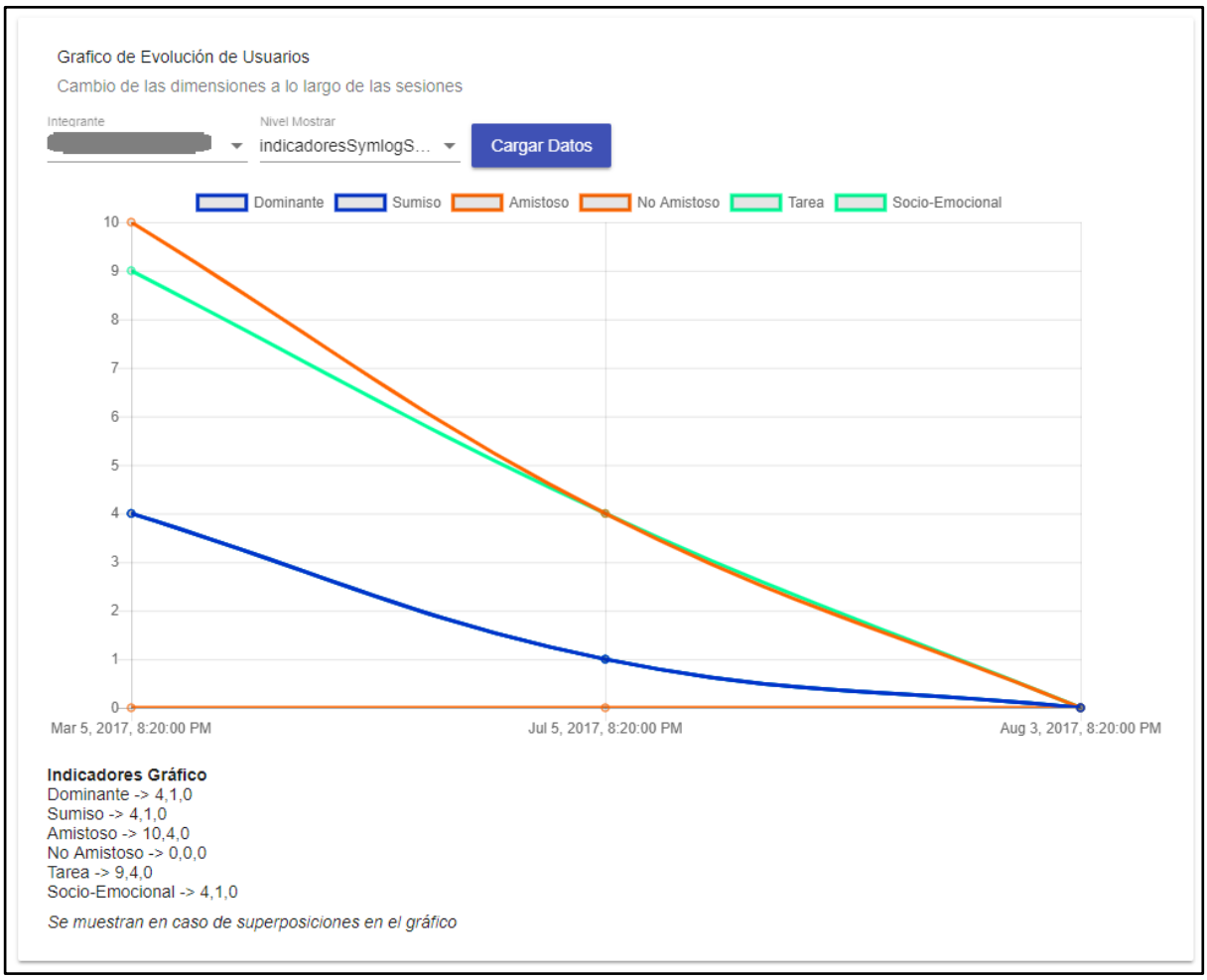

Figura 6.4. Gráfico con los indicadores Symlog que luego generan las respectivas dimensiones

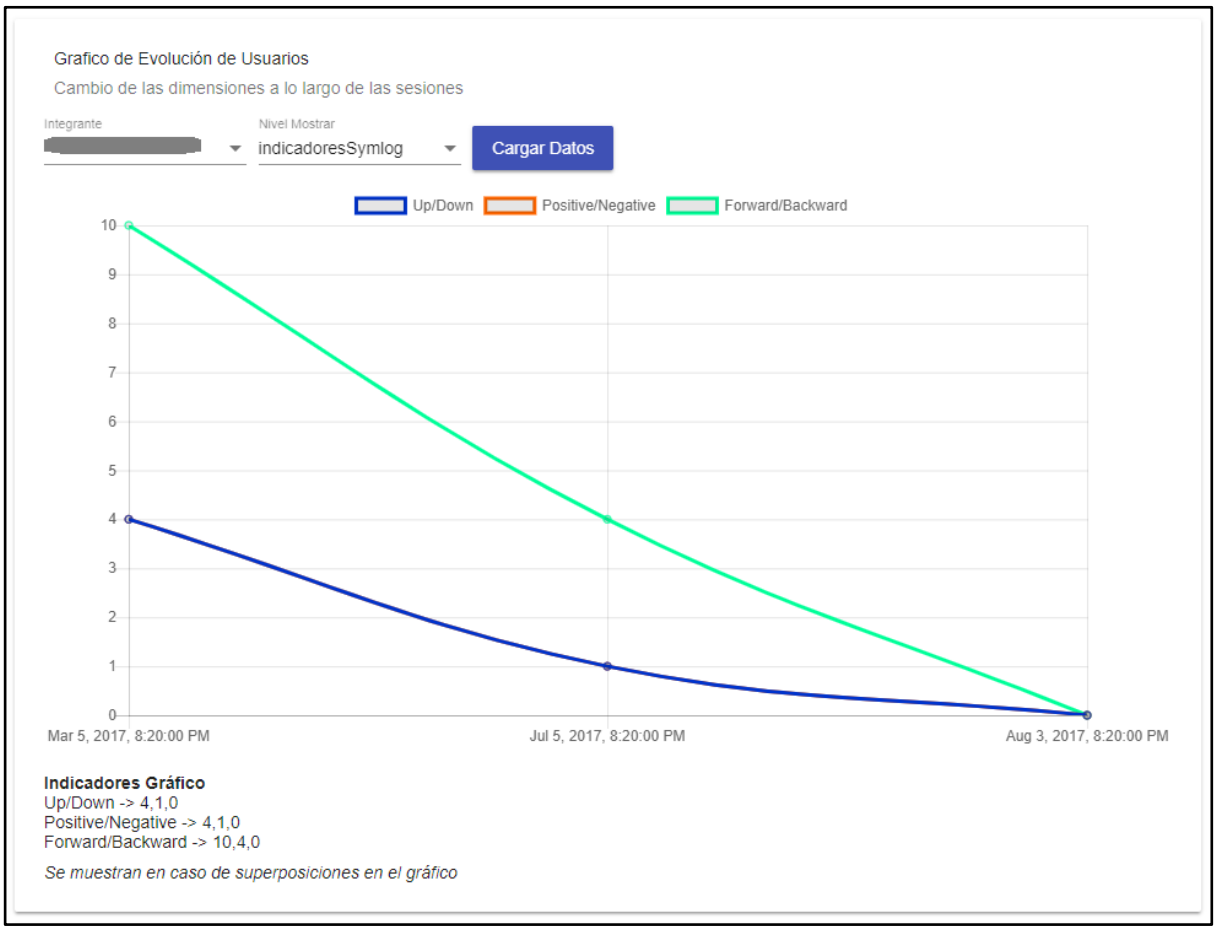

Figura 6.5. Gráfico con las dimensiones Symlog, producto de agrupar los seis indicadores expuestos arrojados por el framework.

Como se puede observar en las dos últimas figuras, las tres dimensiones Symlog se generan a partir de los seis indicadores que retorna el framework. Por ejemplo, los indicadores de “Amistoso” y “No Amistoso”, al restarse uno con el otro, generan la dimensión P/N, que en el caso del ejemplo expuesto en la Figura 6.5, coincide que se superpone con la dimensión U/D, en cuanto a valores. En caso de que eso ocurriese, debajo del gráfico se incluyeron los valores de cada dimensión en cada punto. Luego en la Figura 6.4, al haber una superposición total en los dos indicadores que constituyen una dimensión, al momento de visualizar la dimensión correspondiente en la figura siguiente, la misma da cero, ya que se dio una situación en que un indicador se canceló con otro.

En casos en donde haya usuarios con pocos mensajes, se va a dar un caso en donde los indicadores que se muestran en la herramienta sean pobres. Esto se debe a que la baja cantidad de mensajes no permite que los indicadores correspondientes no sean lo suficientemente robustos como para tenerse en cuenta. Dicho de otra manera y a modo de ejemplificación, si se quiere analizar el perfil de una persona que tuvo solo diez mensajes en una determinada sesión, si bien se generan indicadores, lo recomendable es no tenerlos en cuenta por falta de información.

A continuación, a partir de un dataset con una cantidad de mensajes y sesiones muy superior, se muestra otro ejemplo de generación de perfiles. En dicho dataset se muestran ejemplos de predicciones de indicadores de usuario más abultados. En este caso, se tomaron mensajes provenientes de conversaciones dentro de una adaptación digital del juego “El Señor de los Anillos”, desarrollada por Cincunegui et al. [9]. Se tomaron 20 conversaciones para ser sometidas al clasificador, lo cual comprende un total de 1158 mensajes. Luego en la siguiente ejemplificación (mostrada a partir de la Figura 6.10), se toman 100 conversaciones provenientes de la misma distribución, a fin de observar los indicadores predichos para usuarios que hayan estado presente en una mayor cantidad de sesiones o conversaciones. En este último caso, las 100 conversaciones comprenden un total de 8223 mensajes a clasificar.

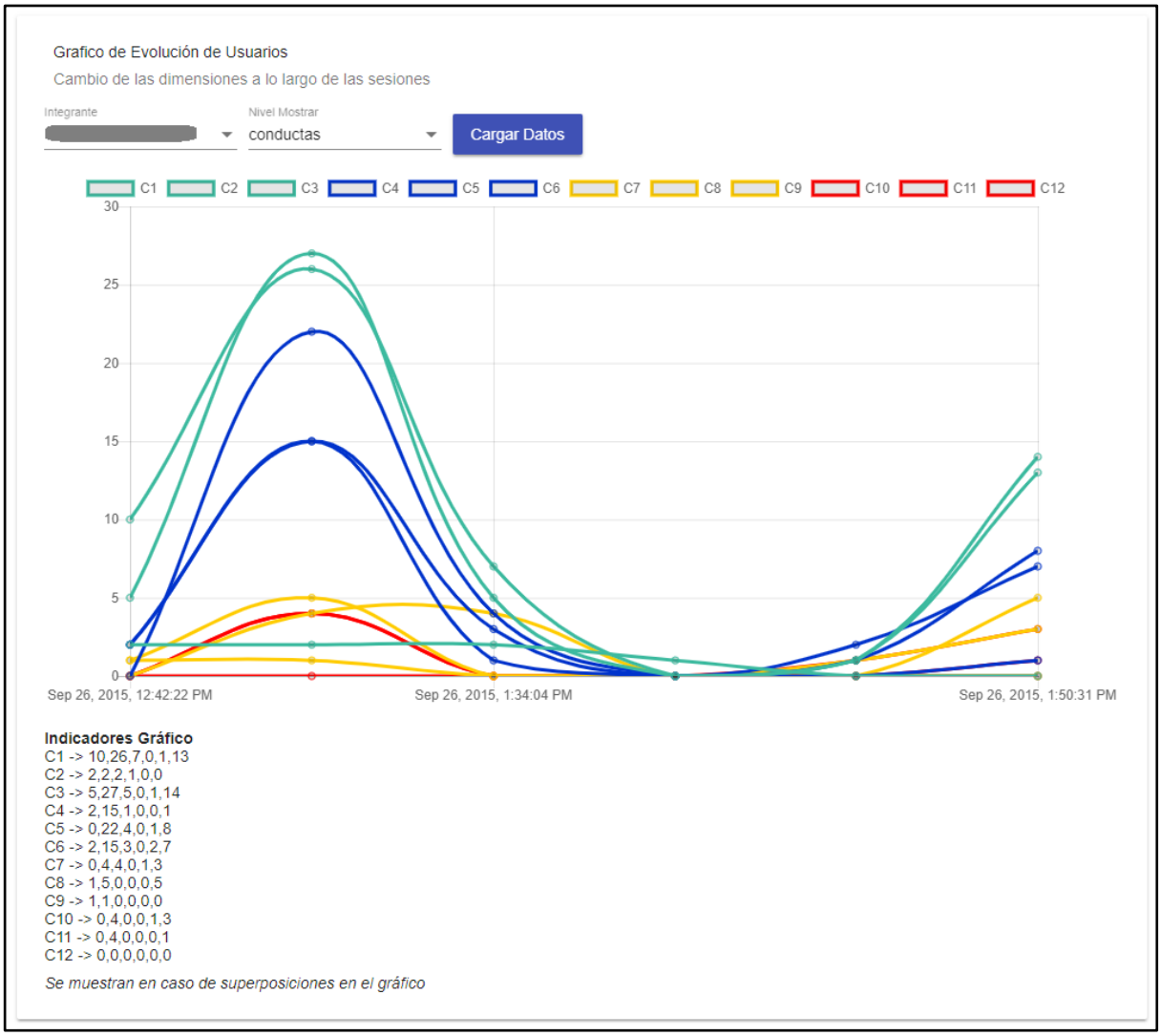

Figura 6.6. Ejemplo de gráfico de conductas a lo largo de las sesiones que tuvo el usuario. En comparación con el ejemplo anterior, ahora se cuenta con el doble de cantidad de sesiones.

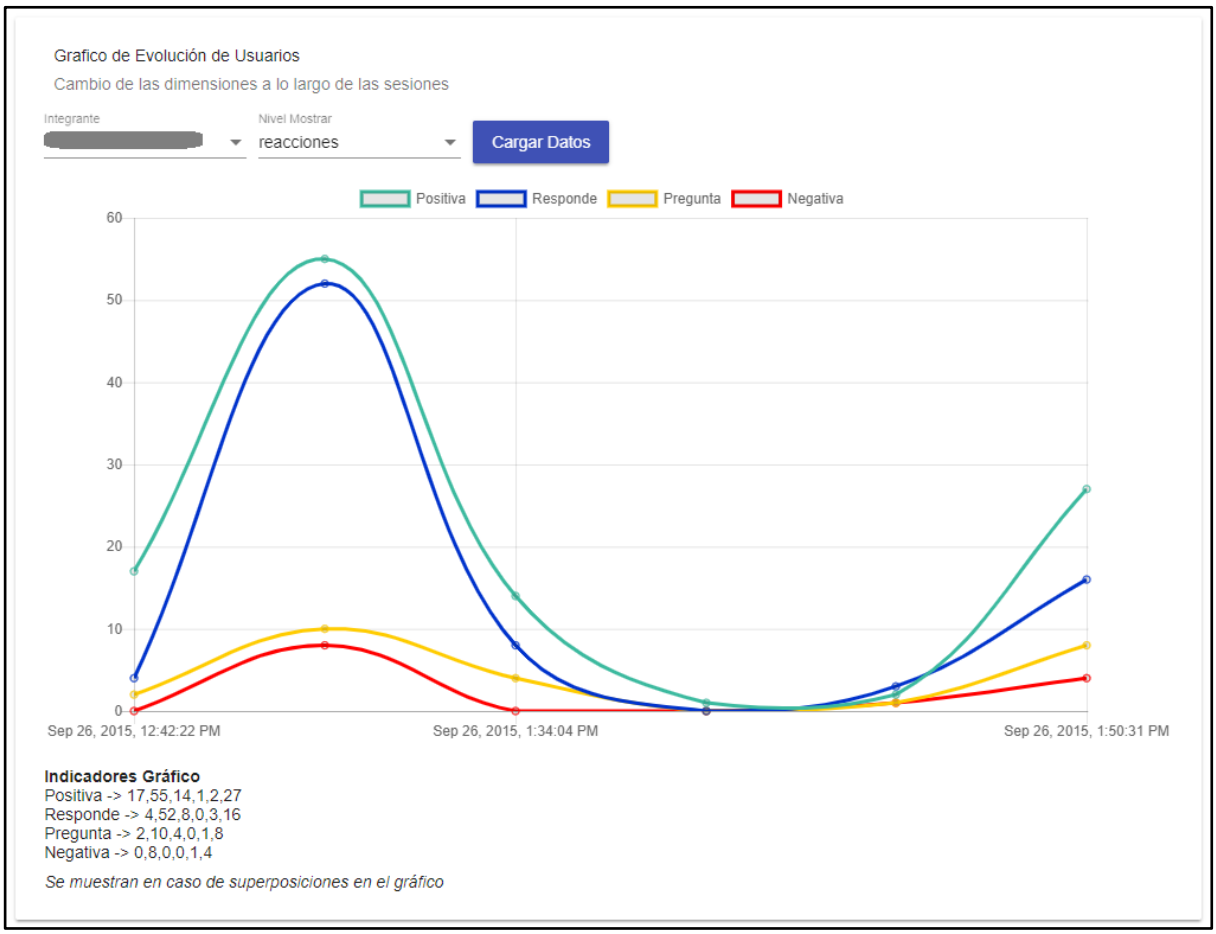

Figura 6.7. Gráfico con las reacciones producto de agrupar las conductas expuestas en la imagen anterior.

A partir de las imágenes para el ejemplo recientemente observado, los indicadores de Symlog que se obtuvieron se muestran a continuación. Primero se muestran los seis indicadores, para luego poder comprender correctamente cómo se fueron armando las tres dimensiones producto de sus indicadores correspondientes.

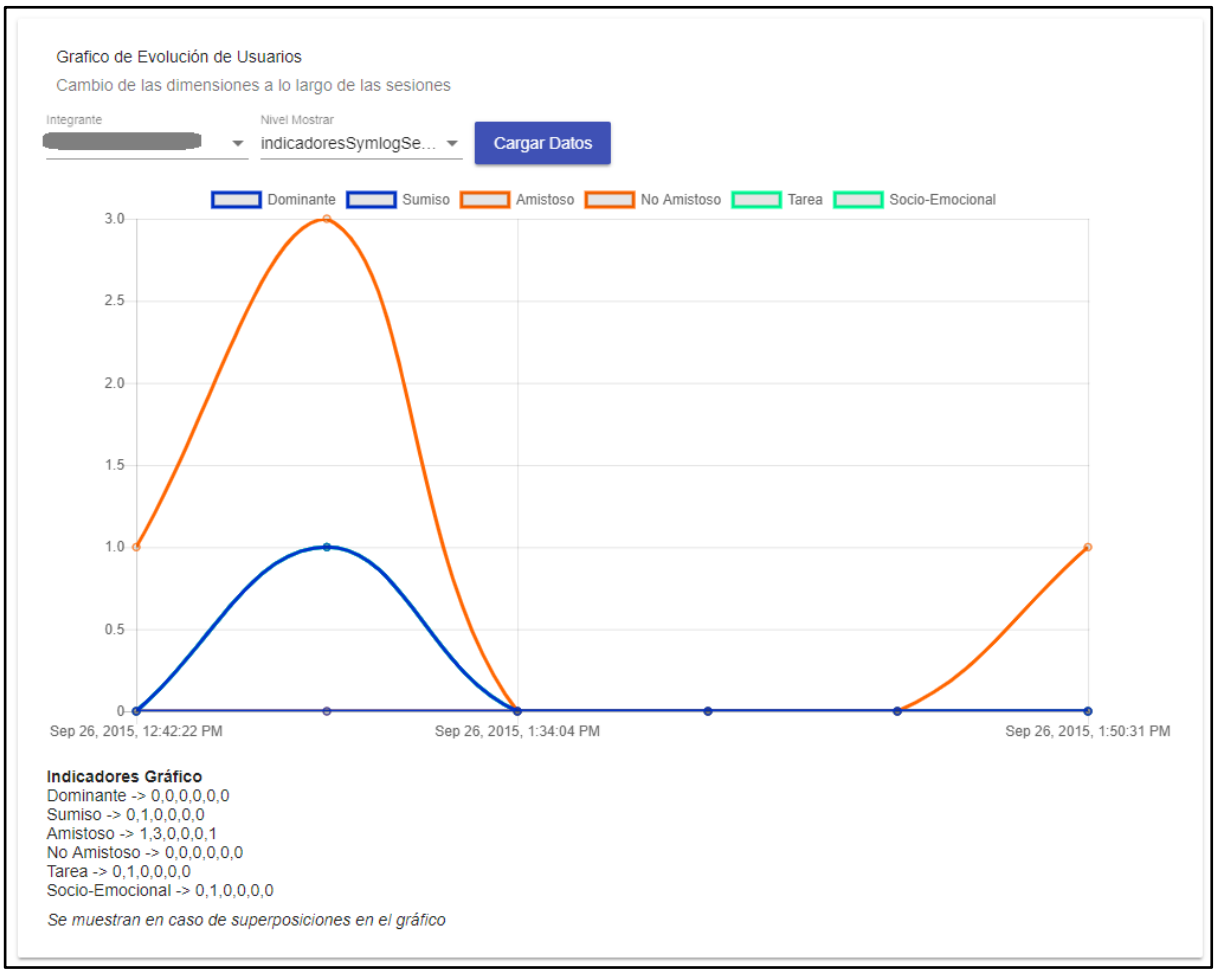

Figura 6.8. Gráfico con los seis indicadores Symlog obtenidos a partir del mapeo de las reacciones.

Figura 6.9. Dimensiones Symlog obtenidas a partir de la agrupación de cada par de indicadores.

Hasta el momento se vieron ejemplos con tres y seis sesiones de trabajo, a partir de los cuales se elaboraron sus respectivos indicadores que en su conjunto comprenden el perfil de usuario generado. A continuación, a partir de la Figura 6.10 se expone un último ejemplo aún más abultado en cuanto a sesiones, a partir de la clasificación de las 50 conversaciones descritas anteriormente. Se muestra un ejemplo de un integrante que participó en diez sesiones de trabajo. Como se puede observar en las últimas figuras, por cuestiones de espacio e implementación, el gráfico no puede mostrar todas las etiquetas de cada punto. Es decir que no se pueden observar todas las marcas de tiempo asociadas a cada observación. Sin embargo, durante el uso de la aplicación, al situar el mouse sobre cualquiera de los puntos, se podrá observar una ventana emergente indicando la marca de tiempo en donde se ubica el punto del gráfico en cuestión.

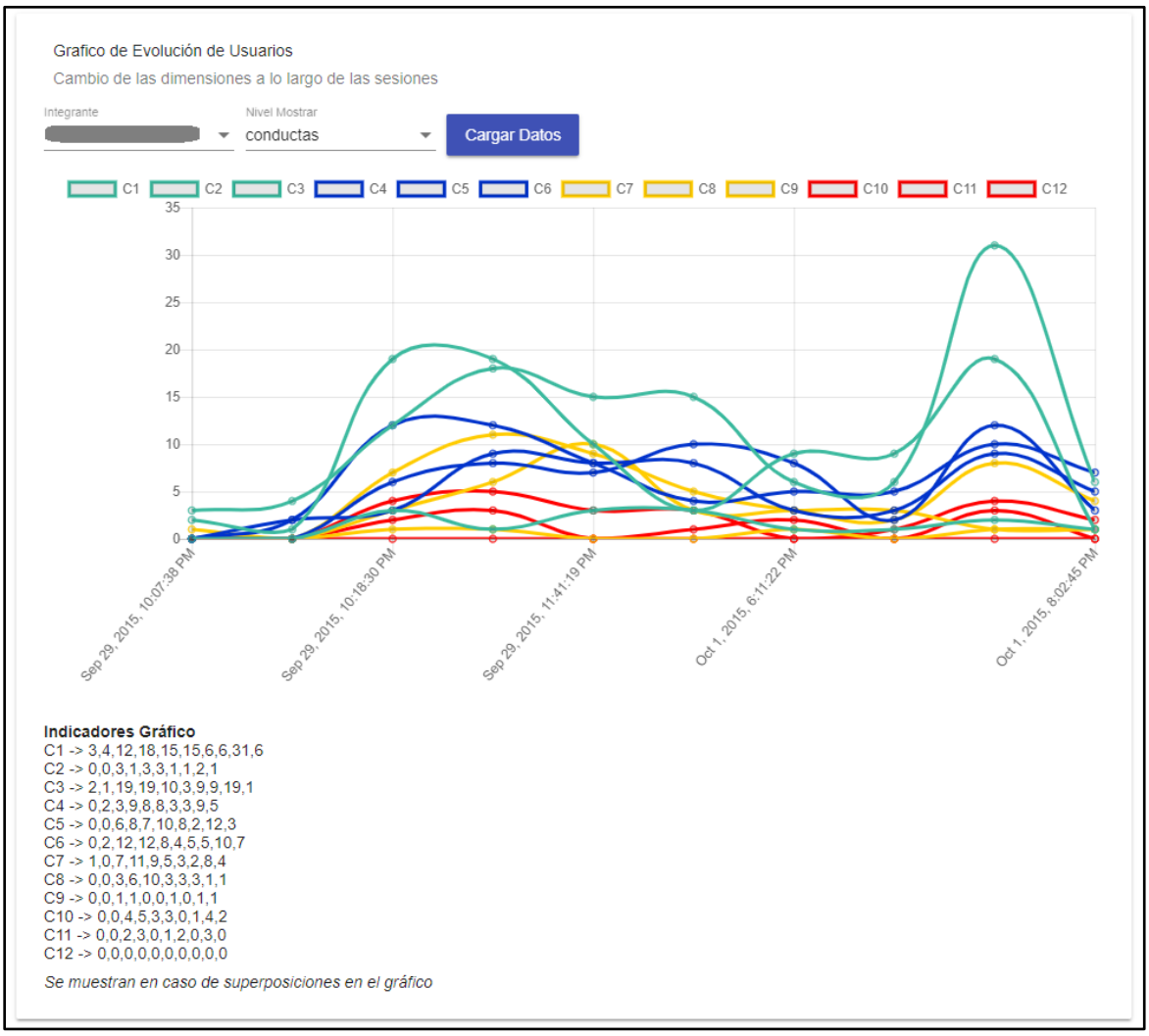

Figura 6.10. Gráfico con las conductas predichas por la red neuronal (CRNN) en las diez sesiones donde el usuario figuró.

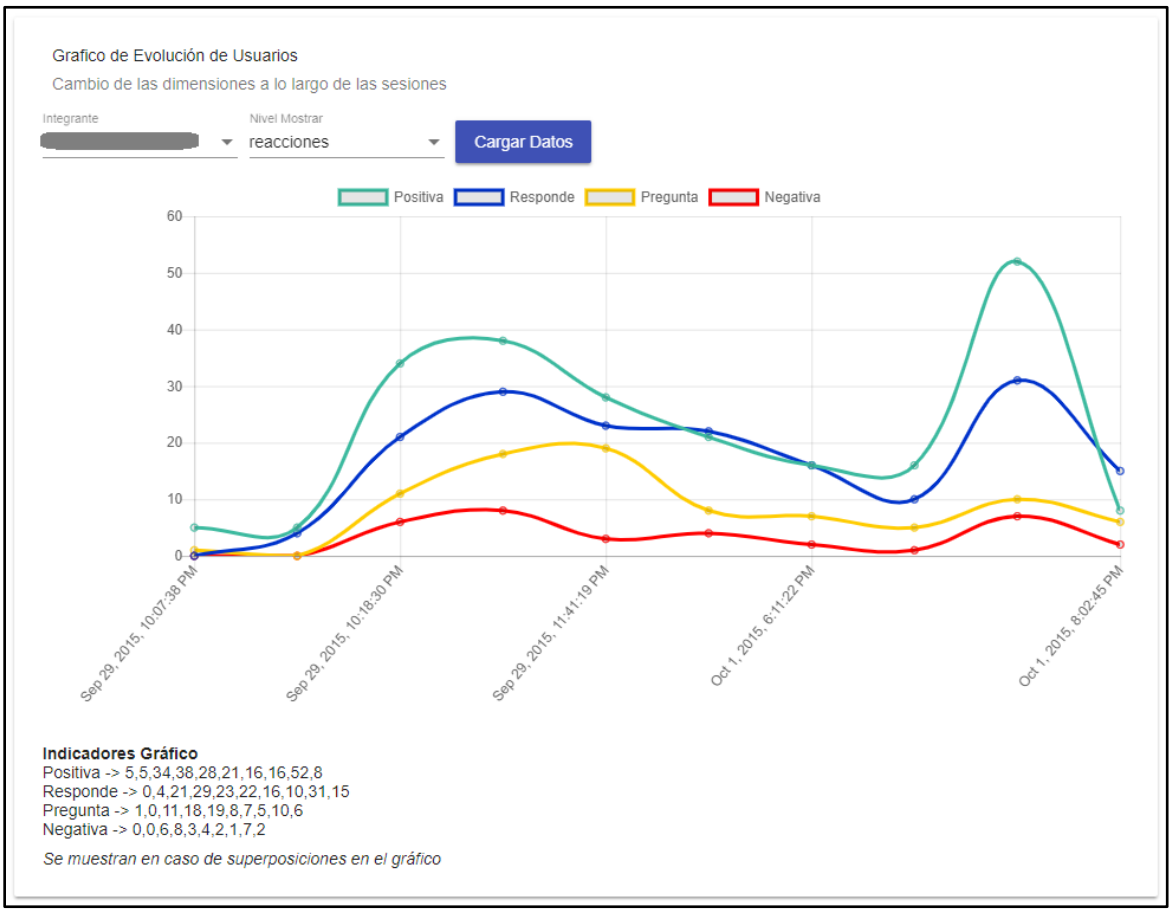

Figura 6.11. Gráfico de reacciones generado a partir de las conductas predichas por la red neuronal.

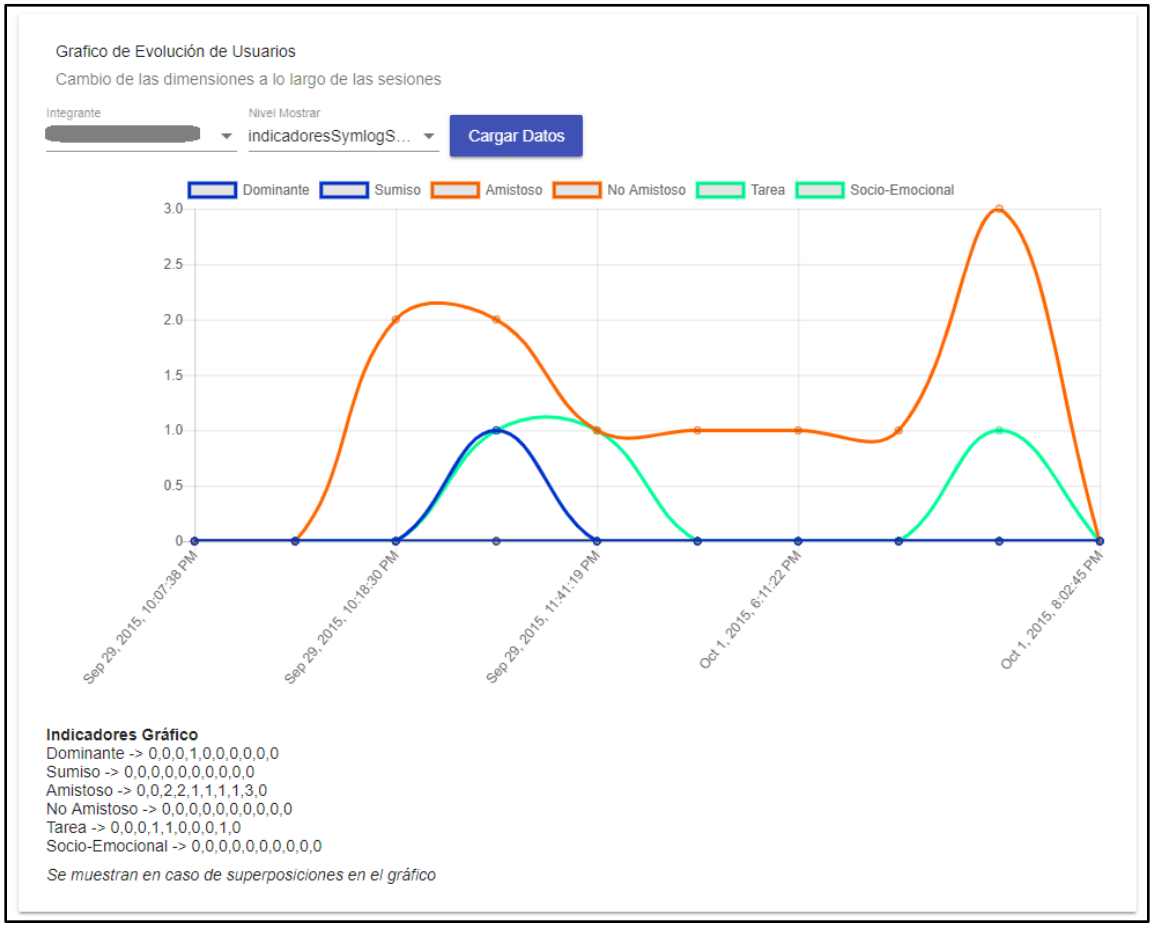

Figura 6.12. Gráfico con los seis indicadores Symlog obtenidos producto del mapeo a partir de las reacciones.

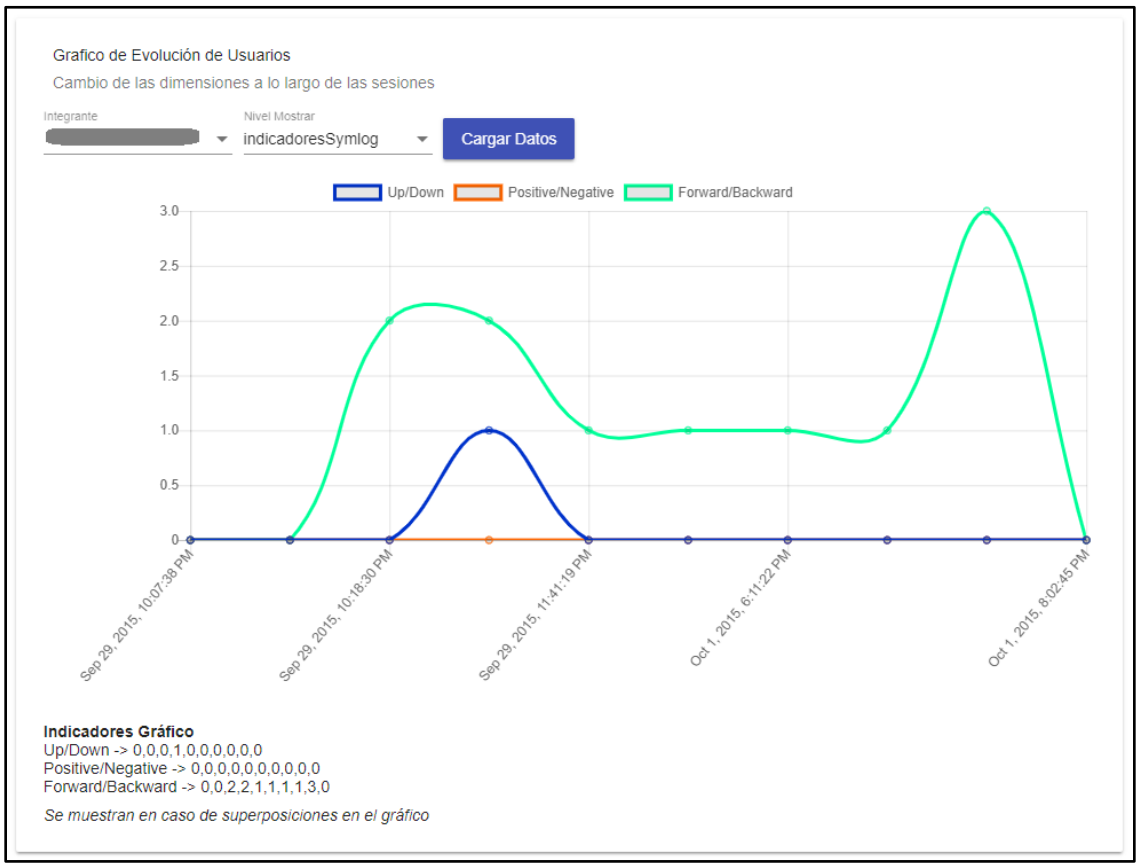

Figura 6.13. Dimensiones Symlog obtenidas a partir de la agrupación de cada par de indicadores.

Hasta el momento se mostraron casos de estudio de los perfiles de usuario asociados a cada persona, pero en cada sesión, es decir, la evolución de su perfil a lo largo del tiempo. Además de estos gráficos, la aplicación web dispone de tablas para cada conjunto de indicadores, agrupándolos por persona. Esto quiere decir que en cada fila se obtiene la suma de los indicadores que cada persona obtuvo en cada sesión. A continuación, se adjuntan imágenes tomadas de la aplicación en donde se observa, para cada ejemplo expuesto, los indicadores asociados.

Tablas para individuo que participó en 3 sesiones

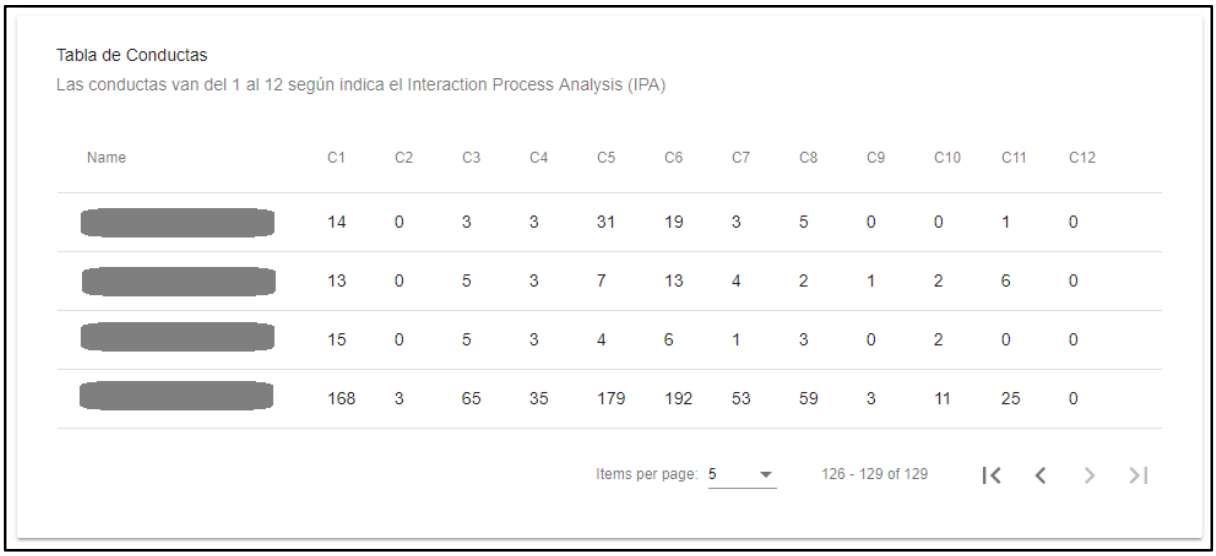

Tabla de Conductas

Las conductas van del 1 al 12 según indica el Interaction Process Analysis (IPA)

Name	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
[redacted]	14	0	3	3	31	19	3	5	0	0	1	0
[redacted]	13	0	5	3	7	13	4	2	1	2	6	0
[redacted]	15	0	5	3	4	6	1	3	0	2	0	0
[redacted]	168	3	65	35	179	192	53	59	3	11	25	0

Items per page: 5 126 - 129 of 129

Figura 6.14. Tabla con los indicadores de conducta agrupados de todas las sesiones para cada usuario.

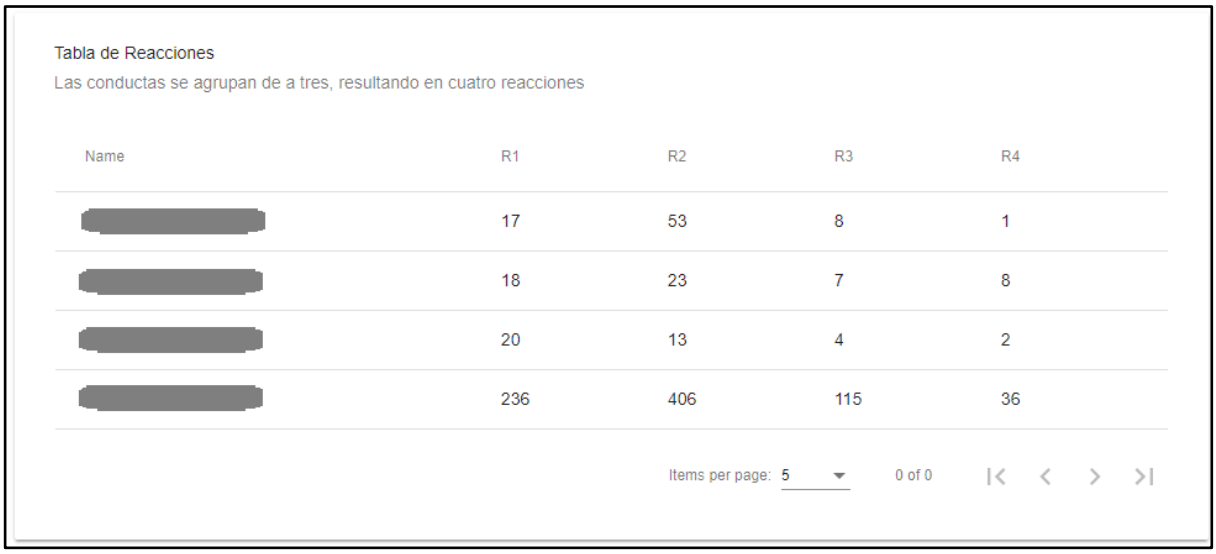

Tabla de Reacciones

Las conductas se agrupan de a tres, resultando en cuatro reacciones

Name	R1	R2	R3	R4
[redacted]	17	53	8	1
[redacted]	18	23	7	8
[redacted]	20	13	4	2
[redacted]	236	406	115	36

Items per page: 5 0 of 0

Figura 6.15. Tabla con los indicadores de reacciones agrupados de todas las sesiones para cada usuario.

Tabla de Symlog

Los indicadores Symlog se obtienen a través de un mapeo de las reacciones

Name	Dominante	Sumiso	Amistoso	No Amistoso	Tarea	Socio Emocional
[redacted]	0	0	1	0	1	0
[redacted]	0	0	1	0	0	0
[redacted]	0	0	1	0	0	0
[redacted]	5	5	14	0	13	5

Items per page: 5 0 of 0

Figura 6.16. Tabla con los indicadores de dimensión Symlog agrupados de todas las sesiones para cada usuario.

Tablas para individuo que participó en 6 sesiones

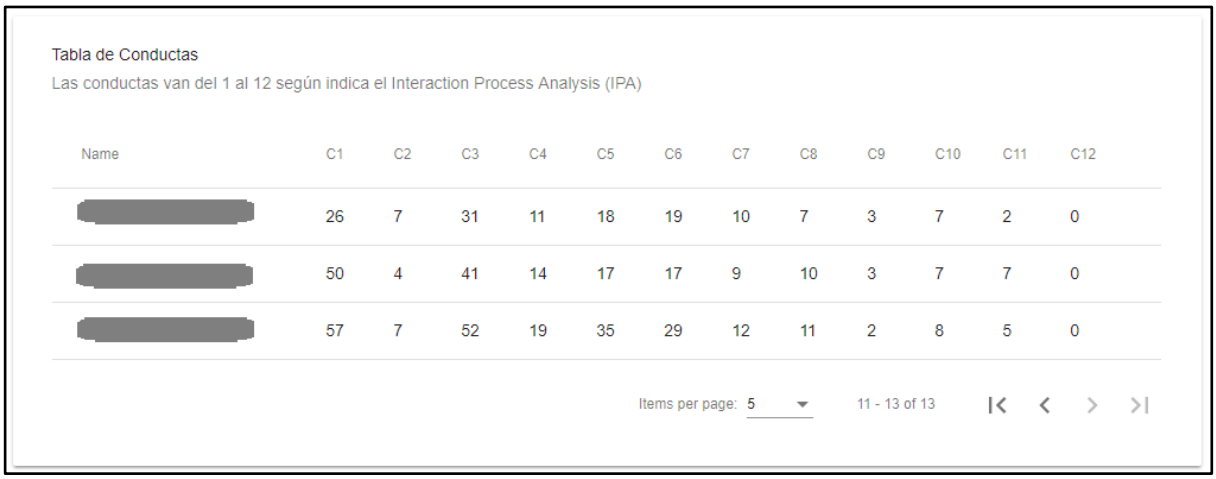

Tabla de Conductas

Las conductas van del 1 al 12 según indica el Interaction Process Analysis (IPA)

Name	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
[redacted]	26	7	31	11	18	19	10	7	3	7	2	0
[redacted]	50	4	41	14	17	17	9	10	3	7	7	0
[redacted]	57	7	52	19	35	29	12	11	2	8	5	0

Items per page: 5 11 - 13 of 13

Figura 6.17. Tabla con los indicadores de conducta agrupados de todas las sesiones para cada usuario.

Figura 6.18. Tabla con los indicadores de reacción agrupados de todas las sesiones para cada usuario.

Figura 6.19. Tabla con los indicadores de dimensión Symlog agrupados de todas las sesiones para cada usuario.

Tabla para Individuo que participó en 10 sesiones

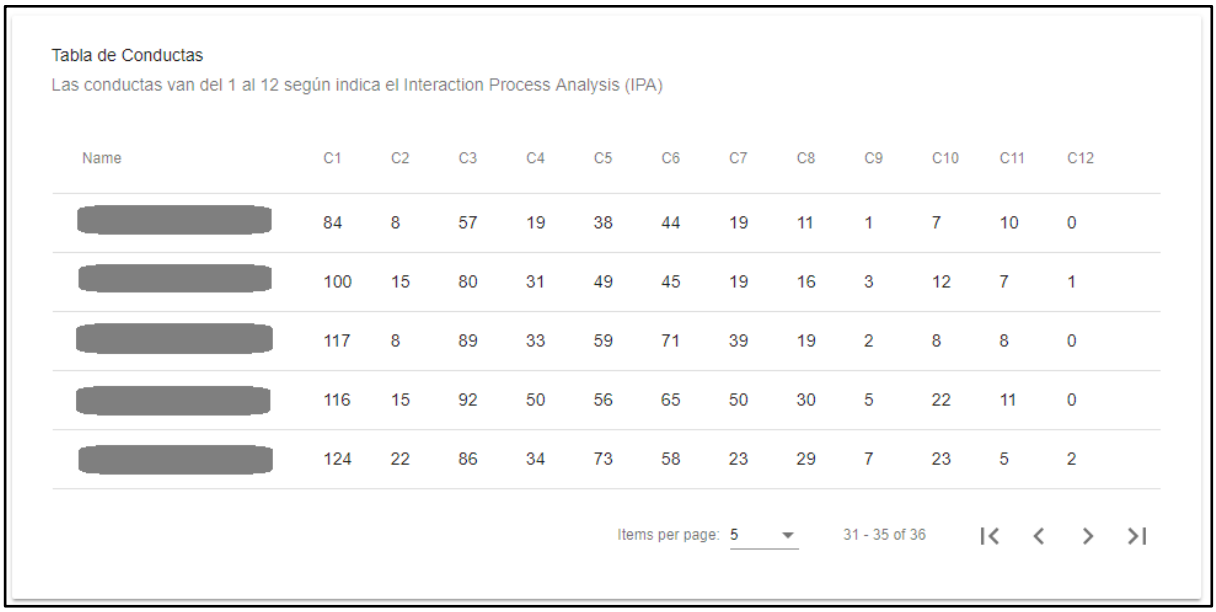

Tabla de Conductas

Las conductas van del 1 al 12 según indica el Interaction Process Analysis (IPA)

Name	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
[redacted]	84	8	57	19	38	44	19	11	1	7	10	0
[redacted]	100	15	80	31	49	45	19	16	3	12	7	1
[redacted]	117	8	89	33	59	71	39	19	2	8	8	0
[redacted]	116	15	92	50	56	65	50	30	5	22	11	0
[redacted]	124	22	86	34	73	58	23	29	7	23	5	2

Items per page: 5 31 - 35 of 36

Figura 6.20. Tabla con los indicadores de conducta agrupados de todas las sesiones para cada usuario.

Tabla de Reacciones

Las conductas se agrupan de a tres, resultando en cuatro reacciones

Name	R1	R2	R3	R4
[redacted]	149	101	31	17
[redacted]	195	125	38	20
[redacted]	214	163	60	16
[redacted]	223	171	85	33
[redacted]	232	165	59	30

Items per page: 5 0 of 0

Figura 6.21. Tabla con los indicadores de reacciones agrupados de todas las sesiones para cada usuario.

Figura 6.22. Tabla con los indicadores de dimensión Symlog agrupados de todas las sesiones para cada usuario.

6.4. Conclusión

En este capítulo se trataron diversos aspectos que van desde el dataset empleado para el desarrollo del framework, hasta el proceso llevado a cabo para realizar la experimentación con dicho framework, junto con los resultados obtenidos.

En la primer sección se abordaron los detalles inherentes a las características de dataset empleado para realizar el entrenamiento de los modelos predictivos. Un correcto entendimiento de las características y del contexto del cual provienen los datos, contribuye a que se pueda realizar clara y correctamente la explicación de la finalidad del framework y herramienta empleada. A partir de las características del dataset, se detalló el proceso llevado a cabo para limpiar el dataset, a fin de maximizar las métricas arrojadas durante el entrenamiento de las redes neuronales. Se utilizaron desde técnicas de data augmentation hasta procedimientos de balanceo de clases, lo cual permitió que el dataset sea más "maleable" para su posterior ajuste por los diversos enfoques de arquitecturas de redes neuronales.

En la segunda sección se procedió de lleno con la experimentación, es decir con la obtención de los indicadores que en su conjunto producen los perfiles de usuario. La experimentación realizada se desarrolla no con la aplicación web, sino con la información cruda que arroja el framework. A partir de las encuestas con los valores Symlog de referencia, se someten los mensajes de los usuarios presentes en las encuestas, a los modelos predictivos. De esta manera se comparó a los perfiles generados por el framework, contra los perfiles que arrojan las encuestas. El experimento fue realizado con cada uno de los cuatro enfoques, en donde el más complejo de los cuatro, es decir aquél que combina redes convolucionales con recurrentes, obtiene el mayor valor de accuracy, siendo de 0.804. Por detrás de este enfoque se encontró sorprendentemente al enfoque de redes

convolucionales, relegando al enfoque de redes recurrentes al tercer puesto, mientras que el enfoque más básico empleando unidades densas quedó relegado cómodamente al cuarto puesto. A continuación, se provee una tabla con la comparación del accuracy para cada enfoque.

Enfoque	Accuracy
Redes Neuronales Planas	0.682
Redes Neuronales Convolucionales (CNN)	0.764
Redes Neuronales Recurrentes (RNN)	0.738
Redes Neuronales Convolucionales Recurrentes (CRNN)	**0.804**

Tabla 6.31. Tabla de comparación de los enfoques desarrollados.

Por último, se realizaron experimentos través de la aplicación web desarrollada. Se mostraron ejemplos para casos de integrantes que participaron en distinta cantidad de sesiones cada uno. Para cada ejemplo, se mostró su correspondiente gráfico de líneas que indicaba el perfil de usuario en cada sesión de trabajo. Como las sesiones de trabajo fueron dispuestas en orden temporal, el gráfico permite el monitoreo de la evolución del perfil de cada usuario a lo largo del tiempo. Además, se proveen las tablas correspondientes, en donde se muestra el perfil global de cada persona, resultante de los perfiles que obtuvo en cada sesión.

Esta serie de tablas e indicadores que la aplicación web provee, constituyen el perfil de usuario que cada perfil posee a lo largo de una serie de sesiones de trabajo. Dichos perfiles permiten que un hipotético supervisor o autoridad competente, posea un conocimiento del perfil colaborativo de cada usuario. De esta manera, el proceso de armado de grupos se puede optimizar de forma sustancial, sobre todo en contextos de equipos de trabajo distribuidos, en donde la interacción física es más limitada.

Además, a partir del monitoreo de la evolución de los perfiles a través del tiempo, se pueden detectar conflictos de colaboración, como por ejemplo en gráficos donde se pueda observar un pico de las reacciones negativas. Otro tipo de situaciones que se pueden detectar a través de la herramienta desarrollada, son los perfiles globales. Por ejemplo, al observar que un determinado individuo posee una elevada cantidad de mensajes negativos sostenida a lo largo de todas sus sesiones, se puede inferir que dicho individuo posee un comportamiento colaborativo conflictivo o poco satisfactorio.

Capítulo 7: Conclusiones

En este trabajo de grado se presentó un framework para elaborar perfiles de usuario, empleando técnicas de aprendizaje profundo. Para dicho fin se desarrolló además una aplicación web que emplee dicho framework a fin de proveerle al usuario final una forma amena de visualización de la información de los perfiles. Dicha cuestión se basa en la premisa de que el usuario final no debe provenir necesariamente del área de sistemas, por lo que es muy importante hacer énfasis en gráficos y tablas para poder mostrar la información lo suficientemente clara y pulida para que pueda ser interpretada correctamente.

Como los perfiles de usuario se generan a partir de un conjunto de interacciones provisto por un usuario final, el framework implementado ofrece la posibilidad de trabajar con diversos formatos. De esta manera se limitan las restricciones de formato, a fin de favorecer la variedad de información que se soporta. Luego internamente el framework realiza conversiones de compatibilidad a un formato interno, a fin de unificar los formatos soportados.

A fin de garantizar la independencia de cada funcionalidad (entre otras cosas), se optó por desarrollar la solución empleando una arquitectura orientado a servicios, en donde se tiene un broker encargado de dirigir el funcionamiento del backend. Para la implementación de dicha arquitectura, se empleó tecnología de contenedores, garantizando también otros aspectos como el bajo acoplamiento, encapsulación, independencia de plataforma, facilidad de deployment, etc.

Con respecto al modelo predictivo, se desarrollaron cuatro enfoques de redes neuronales para la generación de los perfiles de usuario. El concepto subyacente que emplea cada enfoque, se complejiza a medida que se entrenaba cada modelo. Es decir que, si el primer modelo emplea los conceptos más básicos, el último utiliza los conceptos más complejos de los cuatro enfoques.

El primer enfoque consistió en el uso de redes neuronales planas, es decir aplicando solamente capas densas. El segundo consistió en redes neuronales convolucionales, en donde la primer etapa se encarga de la selección de características, mientras que la segunda efectúa la clasificación propiamente dicha. Luego el tercer enfoque consistió en redes neuronales recurrentes, basadas en celdas LSTM a fin de detectar relaciones dentro de una misma instancia. Por último, se experimentó con un modelo que mezcla éstos últimos conceptos. Comienza con una sección de convolucionales, para luego atravesar una sección recurrente, para finalmente clasificar con capas densas. Este enfoque de redes suele ser denominado como redes neuronales convolucionales recurrentes (CRNN).

Como era de esperar, este último enfoque fue el que mejor resultados arrojó. Con un accuracy de 0.804, se encontró por delante (de forma sorpresiva) del enfoque basado en redes convolucionales, con el cual se obtuvo un accuracy de 0.764. Por detrás se encontró el enfoque de recurrentes, que obtuvo un accuracy de 0.738, mientras que el más básico de los cuatro, anduvo muy alejado, con un valor de 0.682.

7.1. Implicancias

El trabajo desarrollado, provee diversas contribuciones para mejorar la forma en que se desarrolla un determinado trabajo colaborativo. Dichas contribuciones giran en torno a tres ejes: el primero relacionado al armado de grupos, el segundo en torno a la detección de situaciones conflictivas en grupos de colaboración, y el tercero a las posibilidades de extensión de la presente tesis de grado.

Dentro de la aplicación web desarrollada, a partir de un conjunto de mensajes, se proveen los perfiles de usuario para los integrantes presentes en el conjunto provisto. Dichos perfiles de usuario capturan la conducta colaborativa del individuo en cuestión, posibilitando un armado eficiente de grupos de trabajo. Esto es de especial relevancia en las organizaciones ya que, en caso de que se disponga de un grupo de trabajo que funcione de forma eficiente debido a las características de cada uno de los integrantes, el trabajo final resultante, por caracter transitivo, sea mejor. De esta manera, un mejor resultado en el trabajo realizado por un grupo, se traduce en mejores resultados para la organización a la cual pertenece dicho grupo. Esta cuestión es de especial relevancia en el ámbito de las empresas de software, en donde es común que se dispongan de equipos de trabajos distribuidos. En este tipo de casos, la interacción física con cada uno de los individuos es más limitada, por lo que la captura de los perfiles de usuario a través de la herramienta desarrollada, constituye ser una ayuda de gran relevancia para los supervisores o autoridades competentes.

El segundo eje sobre el cual se efectúan contribuciones en el presente trabajo de grado, es la detección de situaciones conflictivas. Naturalmente, una situación conflictiva en un grupo de colaboración se asocia con perfiles de usuario negativos. Los perfiles de usuario que la herramienta muestra, contemplan interacciones negativas, las cuales se manifiestan tanto a través de gráficos como de tablas. De esta manera, el usuario final, al observar este tipo de cuestiones, podrá tomar conocimiento de los conflictos, a fin de poder tomar las medidas correspondientes. Por ejemplo, al observar en el gráfico un pico en las reacciones negativas de un determinado individuo, indicará un caso en donde se presentó conflictividad en un grupo. En caso de observar que el pico se repite en varias sesiones de trabajo para un mismo individuo, quiere decir que el individuo en cuestión posee una conducta colaborativa conflictiva.

El tercer eje está más relacionado con la implementación del framework. A partir de la aplicación web desarrollada, que utiliza el framework subyacente, se provee la posibilidad de que el usuario escoja dentro de un conjunto variado de formatos, a saber: ARFF, JSON (Takeout), CSV, y a través de una base de datos (MongoDB). De esta manera, el usuario final no debe atarse a un formato específico, sino que se le ofrece flexibilidad. Además, dicha flexibilidad se tuvo en cuenta al momento de diseñar la arquitectura. El enfoque orientado a servicios, permitió la división de la funcionalidad de forma tal que, en caso de que se necesite el soporte de más formatos de mensajes, se tenga que desarrollar solamente el servicio encargado de dicha tarea. Luego solamente se debe modificar la configuración del broker (URL) para incorporar el servicio nuevo al sistema.

7.2. Limitaciones

En líneas generales, las condiciones de desarrollo del trabajo y las premisas sobre las cuales se planteó el trabajo de grado, permitieron llevar a cabo un desarrollo exitoso tanto del framework, como de la aplicación web. La elección de las tecnologías a utilizar fue crucial para facilitar varias cuestiones como por ejemplo el tiempo de desarrollo, la facilidad del deployment, entre otras cuestiones. Sin embargo, cabe mencionar algunos aspectos que constituyeron desafíos interesantes durante el trabajo de grado.

La herramienta de clasificación sobre la cual se desarrolló uno de los servicios que componen al backend, tienen una importante restricción de plataforma debido que utilizan archivos DLL. Dichos archivos consisten en rutinas que pueden ser ejecutadas por aplicaciones, que son cargadas en tiempo de ejecución. Si bien se trata de un buen recurso para disminuir el tamaño final del archivo ejecutable, el gran problema que conlleva es que son una característica restringida al sistema operativo Windows. Las bibliotecas de enlace dinámico son un concepto que prácticamente existe en todos los sistemas operativos, pero en este caso, la denominación DLL es exclusiva de Windows. De esta manera, si bien el ambiente de desarrollo del trabajo de grado fue el mismo sistema operativo, se toma consciencia de que, en un hipotético ambiente de producción, se debe eliminar parte de la funcionalidad que emplea DLLs debido a que, en dichos ambientes, el sistema operativo por defecto es Linux. Este tipo de decisión se dio debido al uso de una librería de NLP llamada FreeLing [48], que por motivos de performance está desarrollada en C++. Como se ofrecen los archivos fuentes de dicha biblioteca, se genera la necesidad de compilar dichos archivos, en donde cada desarrollador debe compilarlos para la plataforma en la cual desarrolla, empleando herramientas específicas, cuyo uso no es el más intuitivo.

Sin embargo, FreeLing es un trabajo que otorga muchas posibilidades debido a los años de trabajo que tiene por detrás. Es una biblioteca muy pulida, y que soporta muchos lenguajes, y que en caso de poderse utilizar fácilmente (es decir reduciendo operaciones burocráticas) a través de lenguajes que hoy son muy utilizados como por ejemplo Python, convertiría a FreeLing en una seria alternativa a otras tecnologías como por ejemplo Stanford CoreNLP [49], que provee mayores facilidades a nivel facilidad de uso.

7.3. Trabajos Futuros

A partir de la implementación llevada a cabo, se pudieron detectar diversos puntos de expansión que pueden derivar en futuros trabajos de grado, o bien líneas de investigación. Se mencionan a continuación los puntos detectados.

7.3.1. Enriquecimiento de Interacciones

El trabajo realizado, consisten en la elaboración de indicadores que conforman perfiles de usuario, a partir de las interacciones de cada usuario en grupos de colaboración. Dichas interacciones consisten en mensajes, que, si bien constituyen el aspecto central de la comunicación, los perfiles que se pudiesen generar podrían ser perfeccionados a partir de diversas acciones que los usuarios

realicen. Un ejemplo de ello son las reacciones de Facebook. Si los mensajes que un individuo envía, son asociados con, por ejemplo, los “Me gusta” que recibe, se podrían inferir o ponderar los mensajes textuales, a fin de producir mejores perfiles de usuario. En el presente trabajo de grado si bien se dejó asentado la estructura sobre la cual se estandarizan las reacciones independientemente de la fuente, la influencia que ejercen sobre los indicadores finales quedó por fuera del alcance de este trabajo.

7.3.2. Edición en línea de Conjuntos de Mensajes

La aplicación web soporta variedad de fuentes, sobre la cual se intentó cubrir formatos ampliamente utilizados. Sin embargo, si bien se le provee al usuario final facilidad para subir los archivos con los mensajes, dichos archivos deben estar correctamente escritos de antemano. Sería muy interesante poder contar con una herramienta que se integre a la aplicación web desarrollada, en donde se provea una suerte de editor de texto para poder proporcionar más fácilmente los mensajes, editándolos en línea, para luego obtener los perfiles asociados.

7.3.3. Detección Automática de Roles de Equipo

Si bien el énfasis del trabajo de grado es la predicción de las dimensiones Symlog, se experimentó también con los indicadores de los roles de Belbin, aunque sin éxito. Se propusieron mapeos a partir de las conductas IPA, aunque se detectó la necesidad de idear el implementar un framework más complejo, y que esté orientado a roles, para modelar correctamente los roles empleando técnicas de inteligencia artificial. En caso de poderse efectuar dicho mapeo, constituiría una línea de trabajo muy interesante sobre la cual generar perfiles de usuario, que podría acoplarse perfectamente a este trabajo de grado.

Anexo A

En este anexo se exponen las matrices de confusión asociadas a la experimentación relacionada a la predicción de los indicadores Symlog. Dichas matrices constituyen la base sobre la cual se calculan las métricas expuestas para cada enfoque de aprendizaje profundo, durante el Capítulo 6.

A.1. Enfoque 1: Redes Planas

Tablas Dimensión U/D (Up/Down)

(U/D) U →	U	No U
U	5	15
No U	58	49

Tabla A.1. Matriz de confusión binaria asociada a la presencia (o no) del indicador de dominancia. La flecha indica valores predichos.

(U/D) - →	-	No -
-	43	58
No -	21	5

Tabla A.2. Matriz de confusión binaria asociada a la presencia (o no) del indicador de intermedio entre dominancia y sumisión. La flecha indica valores predichos.

(U/D) D →	D	No D
D	0	6
No D	0	121

Tabla A.3. Matriz de confusión binaria asociada a la presencia (o no) del indicador de sumisión. La flecha indica valores predichos.

Tablas Dimensión P/N (Positive/Negative)

(P/N) P →	P	No P
P	89	9
No P	27	2

Tabla A.4. Matriz de confusión binaria asociada a la presencia (o no) del indicador de "amistoso". La flecha indica valores predichos.

(P/N) - →	-	No -
-	2	26
No -	9	90

Tabla A.5. Matriz de confusión binaria asociada a la presencia (o no) del indicador de intermedio entre "amistoso" y "no amistoso". La flecha indica valores predichos.

(P/N) N →	N	No N
N	0	1
No N	0	126

Tabla A.6. Matriz de confusión binaria asociada a la presencia (o no) del indicador de "no amistoso". La flecha indica valores predichos.

Tablas Dimensión F/B (Forward/Backward)

(F/B) F →	F	No F
F	48	9
No F	58	12

Tabla A.7. Matriz de confusión binaria asociada a la presencia (o no) del indicador de "orientación a la tarea". La flecha indica valores predichos.

(F/B) - →	-	No -
-	12	58
No -	9	48

Tabla A.8. Matriz de confusión binaria asociada a la presencia (o no) del indicador de intermedio entre "orientación a la tarea" y "orientación a lo socioemocional". La flecha indica valores predichos.

(F/B) B →	B	No B
B	0	0
No B	0	127

Tabla A.9. Matriz de confusión binaria asociada a la presencia (o no) del indicador de "orientación a lo socioemocional". La flecha indica valores predichos.

A.2. Enfoque 2: Redes Neuronales Convolucionales (CNN)

Tablas Dimensión U/D (Up/Down)

(U/D) U →	U	No U
U	3	17
No U	18	89

Tabla A.10. Matriz de confusión binaria asociada a la presencia (o no) del indicador de dominancia. La flecha indica valores predichos.

(U/D) - →	-	No -
-	81	20
No -	23	3

Tabla A.11. Matriz de confusión binaria asociada a la presencia (o no) del indicador de intermedio entre dominancia y sumisión. La flecha indica valores predichos.

(U/D) D →	D	No D
D	0	6
No D	2	119

Tabla A.12. Matriz de confusión binaria asociada a la presencia (o no) del indicador de sumisión. La flecha indica valores predichos.

Tablas Dimensión P/N (Positive/Negative)

(P/N) P →	P	No P
P	96	2
No P	28	1

Tabla A.13. Matriz de confusión binaria asociada a la presencia (o no) del indicador de "amistoso". La flecha indica valores predichos.

(P/N) - →	-	No -
-	1	27
No -	2	97

Tabla A.14. Matriz de confusión binaria asociada a la presencia (o no) del indicador de intermedio entre "amistoso" y "no amistoso". La flecha indica valores predichos.

(P/N) N →	N	No N
N	0	1
No N	0	126

Tabla A.15. Matriz de confusión binaria asociada a la presencia (o no) del indicador de "no amistoso". La flecha indica valores predichos.

Tablas Dimensión F/B (Forward/Backward)

(F/B) F →	F	No F
F	41	16
No F	46	24

Tabla A.16. Matriz de confusión binaria asociada a la presencia (o no) del indicador de "orientación a la tarea". La flecha indica valores predichos.

(F/B) - →	-	No -
-	24	46
No -	16	41

Tabla A.17. Matriz de confusión binaria asociada a la presencia (o no) del indicador de intermedio entre "orientación a la tarea" y "orientación a lo socioemocional". La flecha indica valores predichos.

(F/B) B →	B	No B
B	0	0
No B	0	127

Tabla A.18. Matriz de confusión binaria asociada a la presencia (o no) del indicador de "orientación a lo socioemocional". La flecha indica valores predichos.

A.3. Enfoque 3: Redes Neuronales Recurrentes (RNN)

Tablas Dimensión U/D (Up/Down)

(U/D) U →	U	No U
U	6	14
No U	35	72

Tabla A.19. Matriz de confusión binaria asociada a la presencia (o no) del indicador de dominancia. La flecha indica valores predichos.

(U/D) - →	-	No -
-	65	36
No -	20	6

Tabla A.20. Matriz de confusión binaria asociada a la presencia (o no) del indicador de intermedio entre dominancia y sumisión. La flecha indica valores predichos.

(U/D) D →	D	No D
D	0	6
No D	1	120

Tabla A.21. Matriz de confusión binaria asociada a la presencia (o no) del indicador de sumisión. La flecha indica valores predichos.

Tablas Dimensión P/N (Positive/Negative)

(P/N) P →	P	No P
P	95	3
No P	28	1

Tabla A.22. Matriz de confusión binaria asociada a la presencia (o no) del indicador de "amistoso". La flecha indica valores predichos.

(P/N) - →	-	No -
-	1	27
No -	3	96

Tabla A.23. Matriz de confusión binaria asociada a la presencia (o no) del indicador de intermedio entre "amistoso" y "no amistoso". La flecha indica valores predichos.

(P/N) N →	N	No N
N	0	1
No N	0	126

Tabla A.24. Matriz de confusión binaria asociada a la presencia (o no) del indicador de "no amistoso". La flecha indica valores predichos.

Tablas Dimensión F/B (Forward/Backward)

(F/B) F →	F	No F
F	44	13
No F	50	20

Tabla A.25. Matriz de confusión binaria asociada a la presencia (o no) del indicador de "orientación a la tarea". La flecha indica valores predichos.

(F/B) - →	-	No -
-	20	50
No -	13	44

Tabla A.26. Matriz de confusión binaria asociada a la presencia (o no) del indicador de intermedio entre "orientación a la tarea" y "orientación a lo socioemocional". La flecha indica valores predichos.

(F/B) B →	B	No B
B	0	0
No B	0	127

Tabla A.27. Matriz de confusión binaria asociada a la presencia (o no) del indicador de "orientación a lo socioemocional". La flecha indica valores predichos.

A.4. Enfoque 4: Redes Neuronales Convolucionales Recurrentes (CRNN)

Tablas Dimensión U/D (Up/Down)

(U/D) U →	U	No U
U	2	18
No U	6	101

Tabla A.28. Matriz de confusión binaria asociada a la presencia (o no) del indicador de dominancia. La flecha indica valores predichos.

(U/D) - →	-	No -
-	93	8
No -	24	2

Tabla A.29. Matriz de confusión binaria asociada a la presencia (o no) del indicador de intermedio entre dominancia y sumisión. La flecha indica valores predichos.

(U/D) D →	D	No D
D	0	6
No D	2	119

Tabla A.30. Matriz de confusión binaria asociada a la presencia (o no) del indicador de sumisión. La flecha indica valores predichos.

Tablas Dimensión P/N (Positive/Negative)

(P/N) P →	P	No P
P	98	0
No P	27	2

Tabla A.31. Matriz de confusión binaria asociada a la presencia (o no) del indicador de "amistoso". La flecha indica valores predichos.

(P/N) - →	-	No -
-	2	26
No -	0	99

Tabla A.32. Matriz de confusión binaria asociada a la presencia (o no) del indicador de intermedio entre "amistoso" y "no amistoso". La flecha indica valores predichos.

(P/N) N →	N	No N
N	0	1
No N	0	126

Tabla A.33. Matriz de confusión binaria asociada a la presencia (o no) del indicador de "no amistoso". La flecha indica valores predichos.

Tablas Dimensión F/B (Forward/Backward)

(F/B) F →	F	No F
F	45	12
No F	41	29

Tabla A.34. Matriz de confusión binaria asociada a la presencia (o no) del indicador de "orientación a la tarea". La flecha indica valores predichos.

(F/B) - →	-	No -
-	29	41
No -	12	45

Tabla A.35. Matriz de confusión binaria asociada a la presencia (o no) del indicador de intermedio entre "orientación a la tarea" y "orientación a lo socioemocional". La flecha indica valores predichos.

(F/B) B →	B	No B
B	0	0
No B	0	127

Tabla A.36. Matriz de confusión binaria asociada a la presencia (o no) del indicador de "orientación a lo socioemocional". La flecha indica valores predichos.

Bibliografía

[1] Romero, R. R., & Saune, S. T. (2002). Grupo: objeto y teoría. Lugar Editorial.

[2] Bales, R. F. (1950). A set of categories for the analysis of small group interaction. American Sociological Review, 257-263.

[3] Bales, R. F., Cohen, S. P., & Williamson, S. A. (1979). SYMLOG: A system for the multiple level observation of groups (Vol. 67). New York: Free Press.

[4] Kim, M. S., Kim, Y. S., & Kim, T. H. (2007, January). Analysis of team interaction and team creativity of student design teams based on personal creativity modes. In ASME 2007 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (pp. 55-67). American Society of Mechanical Engineers.

[5] Löfstrand, P., & Zakrisson, I. (2014). Competitive versus non-competitive goals in group decision-making. Small Group Research, 45(4), 451-464.

[6] Johnson, A., & Taatgen, N. (2005). User modeling. The handbook of human factors in web design, 424-438.

[7] Fischer, G. (2001). User modeling in human–computer interaction. User modeling and user-adapted interaction, 11(1-2), 65-86.

[8] Maisonneuve, J. (1998). La dinámica de los grupos. 11va. Edn. Nueva Visión, Argentina.

[9] Berdun, F. D., Armentano, M. G., Berdun, L. S., & Cincunegui, M. (2018). Building SYMLOG profiles with an online collaborative game. International Journal of Human-Computer Studies.

[10] Belbin, R. M. (1993). Team roles at work: A strategy for human resource management. zitiert in: Teamarbeit und Teamentwicklung, 321.

[11] Berdun, F. D., & Armentano, M. G. (2018). Modeling Users Collaborative Behavior with a Serious Game. IEEE Transactions on Games.

[12] Berdun, F. D., Armentano, M. G., Berdun, L., & Mineo, M. (2018). Classification of collaborative behavior from free text interactions. Computers & Electrical Engineering, 65, 428-437.

[13] Pianesi, F., Zancanaro, M., Not, E., Leonardi, C., Falcon, V., & Lepri, B. (2008). Multimodal support to group dynamics. Personal and Ubiquitous Computing, 12(3), 181-195.

[14] Rein, G. L. (1991, January). A group mood meter. InSystem Sciences, 1991. Proceedings of the Twenty--Fourth Annual Hawaii International Conference on(Vol. 4, pp. 308-323). IEEE.

[15] Wainer, J., & Braga, D. P. (2001, September). Symgroup: applying social agents in a group interaction system. In Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work (pp. 224-231). ACM.

[16] Ellis, C., Wainer, J., & Barthelmess, P. (2003). Agent--augmented meetings. In Agent supported cooperative work (pp. 27-52). Springer US.

[17] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. nature, 323(6088), 533.

[18] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[19] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[20] Cho, Kyunghyun & van Merriënboer, Bart & Gulcehre, Caglar & Bougares, Fethi & Schwenk, Holger & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. 10.3115/v1/D14-1179.

[21] Mikolov, T., Chen, K., Corrado, G.S., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. CoRR, abs/1301.3781..

[22] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111-3119).

[23] Le, Q., & Mikolov, T. (2014, January). Distributed representations of sentences and documents. In International Conference on Machine Learning (pp. 1188-1196).

[24] Bojanowski, P., Grave, E., Joulin, A. & Mikolov, T. (2017). Enriching Word Vectors with Subword Information. Transactions of the Association for Computational Linguistics, 5, 135--146.

[25] Joulin, A., Grave, E., Bojanowski, P. & Mikolov, T. (2016). Bag of Tricks for Efficient Text Classification (cite arxiv:1607.01759)

[26] Kiperwasser, E., & Goldberg, Y. (2016). Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. Transactions Of The Association For Computational Linguistics, 4, 313-327.

[27] Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL (p./pp. 1746--1751).

[28] Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In Advances in neural information processing systems (pp. 649-657).

[29] Amir, S., Wallace, B. C., Lyu, H., & Silva, P. C. M. J. (2016). Modelling context with user embeddings for sarcasm detection in social media. Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL), pages 167–177,

Berlin, Germany, August 7-12, 2016.

[30] Poria, S., Cambria, E., Hazarika, D., & Vij, P. (2016). A Deeper Look into Sarcastic Tweets Using Deep Convolutional Neural Networks. COLING.

[31] Xiong, W., Wu, L., Alleva, F., Droppo, J., Huang, X., & Stolcke, A. (2018, April). The Microsoft 2017 conversational speech recognition system. In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 5934-5938). IEEE.

[32] Chan, W., Jaitly, N., Le, Q., & Vinyals, O. (2016, March). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on (pp. 4960-4964). IEEE.

[33] Kingma, D. P. & Ba, J. (2014). Adam: A Method for Stochastic Optimization.. CoRR, abs/1412.6980. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[34] Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method. CoRR, abs/1212.5701.

[35] Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, 12(Jul), 2121-2159.

[36] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), 1929-1958.

[37] Ioffe, S. & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.. CoRR, abs/1502.03167.

[38] Serrano, F., Berdun, F. D., & Armentano, M. G. Identificación automática de usuarios conflictivos en grupos de Facebook. En Proceedings de XIX Simposio Argentino de Inteligencia Artificial (ASAI)-JAIIO 47 (CABA, Argentina, 2018).

[39] Fielding, R. (2000). Representational state transfer. Architectural Styles and the Design of Network-based Software Architecture, 76-85.

[40] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016, November). Tensorflow: a system for large-scale machine learning. In OSDI (Vol. 16, pp. 265-283).

[41] Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C. & Zhang, Z. (2015). MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems.. CoRR, abs/1512.01274.

[42] Seide, F., & Agarwal, A. (2016, August). CNTK: Microsoft's open-source deep-learning toolkit. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 2135-2135). ACM.

[43] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., ... & Darrell, T. (2014, November). Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the 22nd ACM international conference on Multimedia (pp. 675-678). ACM.

[44] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., ... & Lerer, A. (2017). Automatic differentiation in pytorch.

[45] Cristian Cardellino: Spanish Billion Words Corpus and Embeddings (March 2016), https://crscardellino.github.io/SBWCE/

[46] Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann.

[47] Mineo, M., Berdun, F. D., Armentano, M. G. (2018). Herramienta inteligente de formación de equipos para la reducción de conflictos en trabajos colaborativos soportados por computadora. Facultad de Cs. Exactas, Universidad Nacional del Centro de la Provincia de Buenos Aires.

[48] Carreras, X., Chao, I., Padró, L., & Padró, M. (2004, May). FreeLing: An Open-Source Suite of Language Analyzers. In LREC (pp. 239-242).

[49] Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., & McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations (pp. 55-60).