Proyecto de Grado
Ingeniería en Computación

Generación de diálogo utilizando aprendizaje por refuerzo y redes neuronales adversarias

Andrés Bello, Matías Sclavi
{andres.bello, matias.sclavi.garcia}@fing.edu.uy

Tutores:
Diego Garat
Guillermo Moncecchi

Facultad de Ingeniería
Universidad de la República

Montevideo — Uruguay
Noviembre, 2020

Resumen

El procesamiento de lenguaje natural (PLN) intenta modelar la capacidad de los seres humanos para comunicarse entre sí. En el marco de esta área, nuestro trabajo tiene por objetivo la generación de diálogo escrito, particularmente entre dos personas de diferente género, mediante el uso de aprendizaje por refuerzo y redes generativas adversarias (GAN). Para lograr este objetivo se construye un corpus que contiene diálogos de películas en inglés, el cual se utiliza para entrenar dos agentes, uno por cada género. Ambos agentes están implementados como una tarea de aprendizaje por refuerzo, donde cada uno está constituido por dos modelos: un generador y un discriminador. El objetivo del generador es producir las respuestas a los diálogos, mientras que el discriminador tiene la tarea de distinguir si la respuesta es generada por un humano o por el modelo. La calidad de las respuestas generadas por los agentes es evaluada utilizando métricas basadas en heurísticas y mediante evaluación humana. Los resultados obtenidos en esta última muestran que 45 % de las respuestas no son distinguidas como provenientes del modelo, mientras que las métricas basadas en heurísticas presentan que los agentes utilizados están por encima de la línea base, implementada utilizando la similitud semántica de las respuestas.

Palabras claves: generación de diálogo, aprendizaje por refuerzo, redes generativas adversarias, redes neuronales recurrentes.

Índice general

Capítulo 1

Introducción

El lenguaje natural se refiere a la capacidad de los seres humanos para comunicarse de forma diaria entre sí, ya sea de forma oral como escrita. El procesamiento de lenguaje natural (PLN) busca crear distintos modelos computacionales del lenguaje natural como, por ejemplo, el diálogo entre personas o la generación de texto. El área de PLN se ha desarrollado cada vez más en los últimos años y hoy en día es posible construir modelos con la tarea de predecir texto, generar traducciones en tiempo real, analizar texto para extraer información relevante, y generar diálogo escrito para mantener una conversación con un humano.

A pesar de los avances en el área, aún sigue siendo un desafío crear modelos que tengan la habilidad de generar texto con contenido, coherencia y un sentido "humano". Cuando el interés es generar respuestas que ocurren en una conversación entre personas, se presentan problemas tales como mantener el contexto o generar respuestas que den pie a continuar con el flujo de la conversación y no sean demasiado cortas o genéricas. Este problema es la motivación de este trabajo.

En el cuadro 1.1 se presenta un ejemplo de un diálogo entre dos personas, en el cual, dada la línea de entrada *«¿Por qué no podemos estar de acuerdo en esto?»* una posible salida correcta es *«Porque estás tomando decisiones por mí.»*; sin embargo, una salida con la respuesta *«No sé.»* sería una respuesta incorrecta por ser demasiado genérica y no aportar continuidad al diálogo.

Diálogo previo: *¿Por qué no podemos estar de acuerdo en esto?*
Posible respuesta: *Porque estás tomando decisiones por mí.*

Cuadro 1.1: Ejemplo de un diálogo.

En trabajos anteriores de generación de diálogo se utilizan modelos neuronales con capacidad de almacenar el contexto, como las redes recurrentes y las *long short-term memory* (LSTM). En particular, estas redes en conjunto con el modelo *sequence-to-sequence* (Seq2Seq)(Sutskever et al., 2014) son capaces de generar texto a partir de un contexto dado. En el trabajo de Sordoni et al. (2015) se observa que estos modelos tienden a generar respuestas demasiado genéricas y repetitivas, por ejemplo, "*I dont't know*" (Serban et al., 2016). Uno de los motivos por los que esto ocurre es por la frecuencia con la que determinadas palabras u oraciones ocurren en el corpus de entrenamiento. Una dificultad a la hora de entrenar un modelo de estas características es lograr detectar y determinar cuándo una oración es buena en términos de naturalidad y de contenido, en el sentido de que sean oraciones relevantes y con las que el diálogo fluya.

Li et al. (2016) proponen utilizar aprendizaje profundo en conjunto con aprendizaje por refuerzo para sortear los problemas anteriores, utilizando un modelo que simule las características de una conversación humana y tenga en cuenta la influencia a largo plazo de las oraciones generadas. La idea es entrenar un modelo neuronal que mejora su rendimiento en base a una recompensa modelada mediante heurísticas.

Para que una oración generada se considere buena debe ser difícil de distinguir si fue la respuesta de un humano o no. Sin embargo, cuando la recompensa se define mediante heurísticas, las respuestas generadas pueden dar como resultado oraciones de mala calidad, ya que las heurísticas se centran únicamente en aspectos puntuales tales como la naturalidad, la información y la coherencia de la respuesta. Es por esto que Li et al. (2017) plantean implementar un modelo utilizando entrenamiento adversario, propuesto por Goodfellow et al. (2014). El objetivo es entrenar dos modelos, uno que genera las oraciones y otro, el discriminador, que clasifica si las oraciones son generadas por un humano o por el modelo. El problema es abordado como una tarea de aprendizaje por refuerzo, en donde la calidad de las oraciones es medida por su capacidad de engañar al discriminador haciéndole creer que la oración fue generada por un humano.

El objetivo del proyecto es el estudio del uso de técnicas de aprendizaje por refuerzo para la generación de diálogos escritos. En particular, se busca la creación de modelos para diálogos entre personas de distintos géneros, para luego observar si existen diferencias notorias entre uno y otro. El trabajo se enfoca en la generación sin centrarse en un dominio particular, buscando crear una respuesta a una secuencia de dos o más oraciones tomadas por entrada. Para esto se requiere un corpus que contenga la diferenciación de género entre los participantes. También es necesario investigar implementaciones ya existentes y técnicas de evaluación para medir la calidad de los diálogos generados.

Luego de estudiar diferentes repositorios se decide utilizar el *Cornell Movie Dialogs Corpus* que contiene diálogos de películas y los metadatos de los personajes, dentro de los cuales se encuentra el género de los personajes. Esto facilita la distinción de conversaciones en las que participan personas de distinto género, y que luego de hacer un procesamiento mínimo esté en condiciones de ser utilizado para la implementación del modelo. El corpus seleccionado solo cuenta con diálogos en inglés, ya que no se encuentra uno con la calidad suficiente tanto en inglés como en español, lo que limita este estudio a esta única lengua.

Dadas las mejoras que propone Li et al. (2017) frente a trabajos anteriores en el área, se decide utilizar este modelo como basa para la experimentación. Por otra parte, la evaluación de las respuestas generadas se realiza mediante métricas basadas en heurísticas y una evaluación humana mediante un formulario en donde el evaluador debe reconocer si una respuesta fue generada por un modelo entrenado o es "real" (esto es, proviene del corpus).

1.1. Cronograma

El trabajo llevó aproximadamente un año y ocho meses. En el cuadro 1.2 se presentan las diferentes actividades realizadas junto con la fecha estimada y la fecha real de realización. Las etapas que provocaron un retraso significativo fueron conseguir y configurar la infraestructura necesaria para poder entrenar los modelos, y lograr ejecutar la implementación de los trabajos seleccionados.

1.2. Estructura del informe

El documento está organizado en cuatro capítulos. En el capítulo 2 se describen en detalle los conceptos que se utilizan en el trabajo. La selección y construcción del corpus utilizado, y la arquitectura del modelo son presentados en el capítulo 3.

Luego, en el capítulo 4 se comentan las distintas formas de evaluación que se utilizan y se presentan los resultados recabados. Finalmente, el capítulo 5 consiste en las conclusiones del trabajo.

Fecha estimada	**Fecha real**	**Actividad**
Marzo 2019 - Abril 2019	Marzo 2019 - Abril 2019	Estudio del estado del arte y definición problema de generación de diálogo entre personajes femeninos y masculinos.
Mayo 2019 - Julio 2019	Mayo 2019 - Agosto 2019	Construcción del corpus. Preparación ambiente de entrenamiento.
Mayo 2019 - Julio 2019	Junio 2019 - Diciembre 2019	Configuración de ambiente de pruebas y ejecución. Pruebas de diferentes implementaciones, entrenamientos, y arquitecturas.
Agosto 2019 - Setiembre 2019	Febrero 2020 - Mayo 2020	Selección de implementación final. Ajustes de implementación. Entrenamiento.
Octubre 2019 - Octubre 2019	Junio 2020 - Junio 2020	Implementación de evaluaciones. Implementación chat web.
Octubre 2019 - Noviembre 2019	Julio 2020 - Noviembre 2020	Recolección y análisis de métricas. Informe final.

Cuadro 1.2: Cronograma de actividades desde el inicio al fin del proyecto.

Capítulo 2

Marco Teórico

Este proyecto consiste en la generación de diálogo escrito mediante el uso de aprendizaje por refuerzo y de la arquitectura redes generativas adversarias. Esta última consiste en la unión de dos modelos, un generador y otro discriminador. El primero tiene la tarea de generar una respuesta dado un diálogo de entrada, mientras que el segundo es un clasificador binario que recibe como entrada un diálogo y debe determinar si fue generado por un humano o no. El generador se entrena para mejorar la calidad de las respuestas en base al resultado que retorna el discriminador, lo que se modela como una tarea de aprendizaje por refuerzo.

En las siguientes secciones se presenta la conceptos de redes neuronales y redes recurrentes (sección 2.1.1), para continuar con redes generativas adversarias (sección 2.2), que forman parte de la arquitectura de este trabajo. Luego se definen los conceptos principales del método de aprendizaje por refuerzo (sección 2.3). Por último se menciona la técnica *Reward for every generation step* (REGS) (sección 2.5) y *Teaching Forcing* (sección 2.6) que se utilizan en el entrenamiento del modelo.

2.1. Redes neuronales

El generador de este trabajo debe retornar, dado un diálogo previo, una oración que sea considerada su continuación. Para ello utiliza el enfoque Seq2Seq (sección 2.1.4), cuyo objetivo es retornar una secuencia de *tokens* dada una secuencia de *tokens* de entrada. Por otro lado, el discriminador se implementa como un clasificador binario que utiliza una estructura LSTM jerárquica para mapear la entrada ($\langle$ *diálogo_previo, respuesta* $\rangle$) a una representación vectorial, y luego determina la probabilidad de que la oración sea generada por una máquina o un humano.

Para mejorar el rendimiento de los modelos LSTM se utiliza el mecanismo *Attention*, que les permite mejorar la relación del contexto con la oración a generar. En la sección 2.1.5 se puede encontrar una explicación del método y, en particular, como lograr mantener la noción del contexto en oraciones largas.

El objetivo de esta sección es introducir las redes neuronales (*feed-forward*) y redes recurrentes, para luego exponer la arquitectura LSTM y el modelo Seq2Seq.

2.1.1. Redes neuronales básicas

Una red neuronal está formada por tres tipos de capas (Mitchell, 1997). La capa inicial es la *capa de entrada* y las neuronas que pertenecen a ellas son llamadas *neuronas de entrada*. La última capa contiene las *neuronas de salida*, y las capas intermedias son denominadas *capas ocultas*, porque las neuronas que la conforman no son ni de entrada ni de salida. La figura 2.1 muestra el esquema de una red neuronal con cuatro capas, donde dos de ellas son capas ocultas.

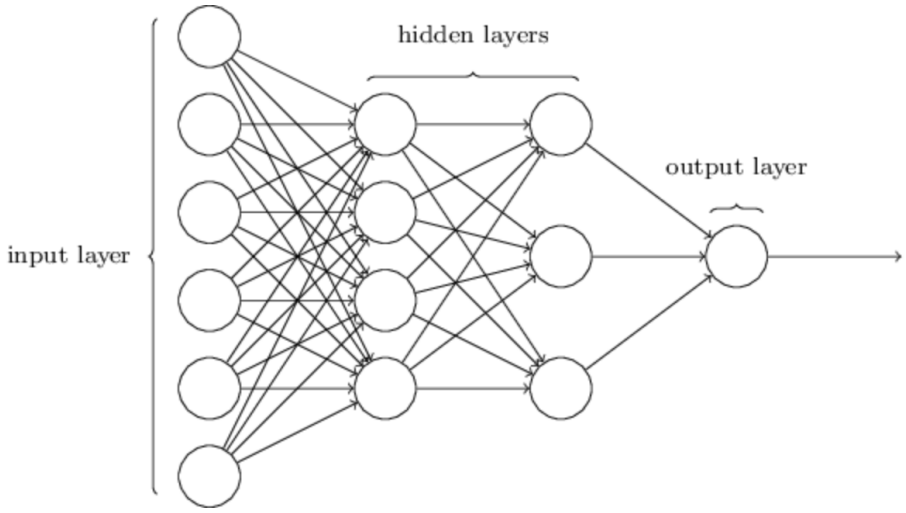

Figura 2.1: Modelo de una red neuronal *feed-forward*.

Este tipo de redes neuronales donde la salida de una capa es la entrada de la capa siguiente se denominan redes *feed-forward*. Esto significa que la red no tiene ciclos y la información es siempre enviada hacia delante, nunca hacia capas anteriores. Las redes que contienen ciclos entre neuronas son las llamadas redes recurrentes, y se presentan en la sección 2.1.2.

Las neuronas que conforman una red neuronal pueden ser de dos tipos: neuronas sigmoides y neuronas perceptrones. Un perceptrón es una función que toma como entrada un vector de valores reales, calcula una combinación lineal y retorna uno si el resultado es mayor a un umbral b o cero en caso contrario. Más precisamente, dada la entrada $\vec{x}$, la salida $o(\vec{x})$ es

$$o(\vec{x}) = \begin{cases} 0 & \text{si } \vec{x} \cdot \vec{w} + b \leq 0 \\ \\ 1 & \text{si } \vec{x} \cdot \vec{w} + b > 0 \end{cases} \tag{2.1}$$

Los perceptrones tiene el problema de que pequeños cambios en los valores w causan grandes alteraciones en la salida de la red o la neurona; sin embargo las neuronas sigmoides frente a cambios pequeños causan pequeños cambios en la salida. Este tipo de neuronas están definidas por

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

El hecho de entrenar una red neuronal se deriva en determinar los valores w_i del vector $\vec{w}$ tal que, para cada ejemplo de entrenamiento, la salida retornada coincida con el valor esperado.

2.1.2. Redes neuronales recurrentes

Las redes neuronales recurrentes (*Recurrent Neural Networks*, RNN) (Mitchell, 1997) son sistemas dinámicos con representaciones de estados temporales. Estas hacen uso de la salida de las neuronas de la red en el tiempo t como la entrada de otras neuronas en $t + 1$, por lo que de esta manera permiten ver a la red como un

grafo dirigido. Son computacionalmente poderosas y pueden ser utilizadas por varias aplicaciones y con varios modelos temporales.

El problema de las RNN básicas es que la información es propagada paso a paso y cuanto más larga es la cadena, más probable es que la información se pierda. Las redes LSTM intentan sortear este problema con el concepto de *celda de estado* o *celda de memoria*.

2.1.3. *Long short-term memory* (LSTM)

Las redes *long short-term memory* (LSTM) son un caso particular de las redes recurrentes, capaces de aprender dependencias a largo plazo. Fueron diseñadas por Hochreiter & Schmidhuber (1997) para resolver el problema de almacenar información en largos períodos de tiempo. A continuación se analiza una neurona para entender la capacidad de almacenar y transmitir información. La figura[1] 2.2 muestra en detalle los componentes de la estructura de la neurona.

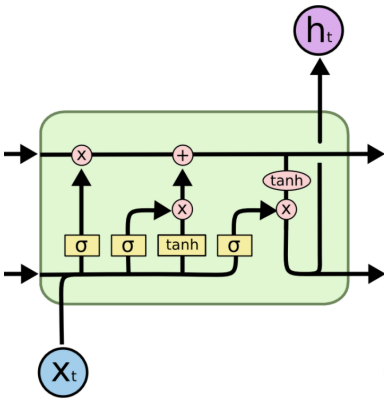

Figura 2.2: Arquitectura de una neurona de una red LSTM.

El concepto principal de una LSTM es la celda de estado o memoria (representada con la línea negra superior en la figura 2.2). La función de esta celda es mantener información relevante que es aprendida a lo largo del tiempo. La red está diseñada para mantener o remover información de esta celda usando tres estructuras diferentes denominadas *puertas*. El objetivo de la primera puerta (representada en la figura 2.3) es decidir qué información del pasado se mantendrá en la celda para propagar hacia delante. Esta decisión está hecha por una capa *sigmoid*.

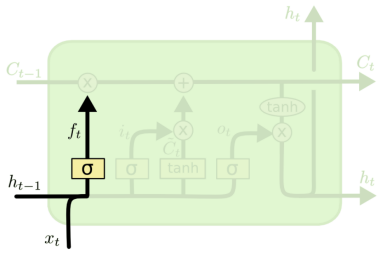

Figura 2.3: Representación de la primera puerta o estructura de una neurona en una red LSTM.

[1]Las figuras 2.2, 2.3, 2.4, 2.5, y 2.6 fueron extraídas de Olah (2015).

La capa recibe como entrada h_{t-1} y x_t (entrada de la red), retornando un valor entre cero y uno por cada elemento de la celda de estado del paso anterior C_{t-1}. Uno representa que el valor de la celda de estado se va a considerar de forma total, mientras que el valor cero significa que no va a ser considerado en absoluto.

El siguiente paso, representado en la figura 2.4, es decidir qué información nueva se almacena en la celda de estado. Esta puerta tiene dos componentes: una capa *sigmoid* para determinar qué valores se actualizan (i_t) y una capa tangente que crea un vector intermedio con los nuevos candidatos a $\tilde{C}_t$.

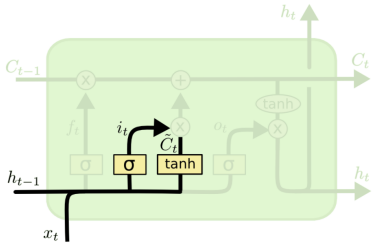

Figura 2.4: Representación de la segunda puerta o estructura de una neurona en una red LSTM.

Luego es necesario actualizar el valor de C_{t-1} para crear la nueva celda de estado C_t. Para ello (figura 2.5) se multiplica el último estado de la celda por f_t y luego se suma $i_t * \tilde{C}_t$.

Por último en la figura 2.6 se muestra como se determina cuál va a ser la salida de la neurona. Para ello, primero se ejecuta una capa *sigmoid* para indicar que parte de la celda de estado se va a eliminar y cuál mantener (análogo a la primera estructura). Luego se crea un vector intermedio de la celda de estado usando una capa tangente y por último se multiplican ambos vectores para generar la salida final.

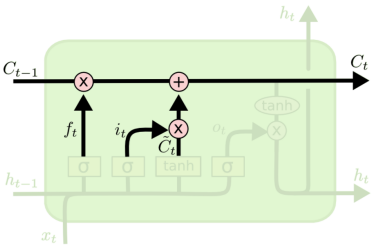

Figura 2.5: Representación de la salida de la segunda puerta o estructura de una neurona en una red LSTM.

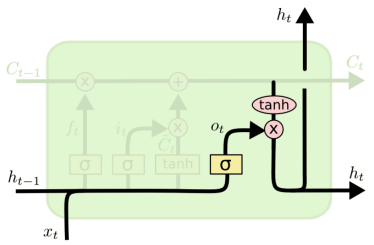

Figura 2.6: Representación de la tercer puerta o estructura de una neurona en una red LSTM.

2.1.4. *Sequence-to-sequence* (Seq2Seq)

En este trabajo se utiliza un modelo neuronal (generador) que debe retornar, dado un diálogo previo representado como una secuencia de palabras, una oración que sea considerada su continuación. Para ello implementa el modelo Seq2Seq propuesto por Sutskever et al. (2014).

El modelo Seq2Seq surge a partir de la necesidad de generar una secuencia de *tokens* S a partir de otra secuencia de *tokens* E, donde S y E pueden tener diferente largo. Este problema se presenta en, por ejemplo, sistemas de traducción de texto, reconocimiento de voz, generación de diálogo, entre otros. Sutskever et al. (2014) proponen resolver este problema utilizando dos LSTM: una (*encoder*) para mapear la secuencia de *tokens* de entrada en una representación vectorial de tamaño fijo, y otra (*decoder*) que utiliza este vector como entrada y una capa *softmax* para generar una secuencia de *token* de salida. En la figura 2.7 se presenta un ejemplo del modelo que recibe como entrada la secuencia "ABC" y retorna la los tokens "WXYZ".

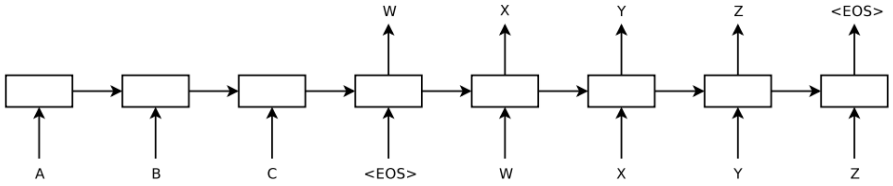

Figura 2.7: Modelo Seq2Seq.

El *encoder* consume de a uno los *tokens* de la secuencia de entrada hasta encontrar el *token* `<EOS>`, que representa el fin de la sentencia. El vector de tamaño fijo que representa la secuencia de entrada está formado por los estados de las neuronas de la última capa oculta del *encoder*. Una vez que el *encoder* procesa el token `<EOS>`, el *decoder* utiliza este vector de tamaño fijo para inicializar los estados de su capa oculta inicial y generar de a uno los *tokens* de salida mediante una función *softmax*. Este proceso continúa hasta generar el *token* `<EOS>` o hasta alcanzar un largo de secuencia máximo.

2.1.5. Mecanismo *Attention*

El mecanismo *attention* está motivado en cómo las personas prestan atención a diferentes regiones de una imagen cuando quieren detectar objetos o cómo se presta atención a determinadas palabras cuando se quiere realizar una traducción (Bahdanau et al., 2015). Por ejemplo, dada la frase *«Ella está comiendo una manzana verde»*, las personas cuando ven la palabra *«comiendo»* saben que pronto va a ocurrir una palabra relacionada a la comida, es este caso *«manzana»*. Por otro lado, la palabra *«verde»* describe la manzana pero probablemente no esté asociada directamente con *«comida»*. *Attention* intenta modelar esta idea utilizando un vector de contexto ponderado por pesos. El objetivo de este vector es representar qué tan fuerte es la correlación de un elemento con el resto.

La necesidad de esta técnica surge porque los modelos básicos de redes neuronales recurrentes, e incluso los modelos LSTM, son incapaces de recordar el contexto de oraciones largas. Generalmente, el comienzo de la oración se va olvidando a medida que es procesada (Bahdanau et al., 2015).

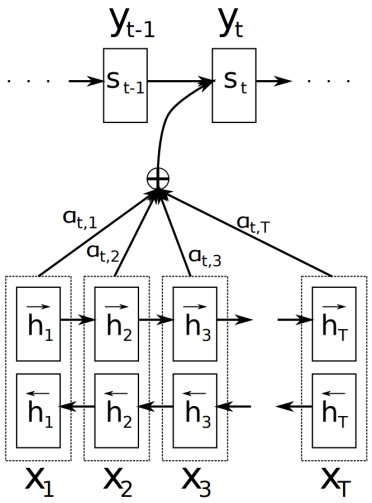

Figura 2.8: Mecanismo de *attention*. Imagen extraída del trabajo de Bahdanau et al. (2015).

El mecanismo de *attention* ayuda a vincular el vector de entrada con el vector de contexto generado por el último estado oculto del *encode*. De esta forma, como se puede ver en la figura 2.8, se crean atajos ponderados por pesos entre la entrada, el contexto y el *decode* asociado.

2.2. Redes generativas adversarias (GAN)

El *framework* propuesto por Goodfellow et al. (2014) enfrenta un modelo generador frente un modelo discriminador, que aprende a determinar si una muestra forma parte de los datos retornados por el generador o del conjunto de datos de entrenamiento. Este *framework* puede utilizar distintos algoritmos de entrenamiento para distintos tipos de modelos y diferentes algoritmos de optimización. En este trabajo, tanto el generador como el discriminador son implementados utilizando redes LSTM.

En el contexto de este trabajo, el generador construye la respuesta a un diálogo intentando engañar al discriminador, el cual tiene la tarea de detectar si la respuesta al diálogo fue generada por el modelo o no. La competencia entre ambos hará que los dos mejoren hasta que lo producido por el generador sea indetectable para el discriminador.

El generador mejora la calidad de las respuestas a partir de los valores que retorna el discriminador para las oraciones generadas hasta el momento; mientras que el discriminador debe recibir tanto respuestas generadas como del conjunto de entrenamiento, para así poder diferenciar entre ellas y dar mejor retroalimentación al generador.

Goodfellow et al. (2014) proponen modelar el entrenamiento como un juego *min-max*[2], en el que participan el generador y el discriminador. De manera intuitiva, el entrenamiento consiste en que el discriminador intenta maximizar la probabilidad de que los datos reales y los datos generados sean clasificados como tales; mientras que el generador intenta minimizar la probabilidad de que el discriminador identifique los datos generados como falsos.

[2]Consiste en minimizar la pérdida en el escenario de pérdida máxima.

2.3. Aprendizaje por refuerzo

Aprendizaje por refuerzo es un método que consiste en modelar cómo un agente, que interactúa con un entorno a través de la selección de acciones determinadas, puede alcanzar una meta maximizando una recompensa que puede ser definida mediante una función (Mitchell, 1997). Esta función se denomina *función recompensa* y retorna un valor $r \in \mathbb{R}$ cada vez que el agente selecciona una acción en un estado determinado. Una forma de definir el problema de maximizar la función recompensa es mediante un *proceso de decisión de Markov* (MDP por sus siglas en inglés).

En un MDP el agente puede estar en un conjunto de estados S de su entorno y tiene un conjunto de acciones posibles A que puede realizar. En cada paso de tiempo t, el agente se encuentra en un estado s_t, selecciona una acción a_t y la ejecuta. Como consecuencia, el entorno responde otorgando al agente una recompensa $r_t = r(s_t, a_t)$ con $r_t \in \mathbb{R}$ y produciendo un cambio de estado a $s_{t+1} = \delta(s_t, a_t)$. En un MDP las funciones $r(s_t, a_t)$ y $\delta(s_t, a_t)$ dependen únicamente del estado y acción actual (propiedad de Markov). En la figura 2.9 se muestra la interacción entre el agente y su entorno.

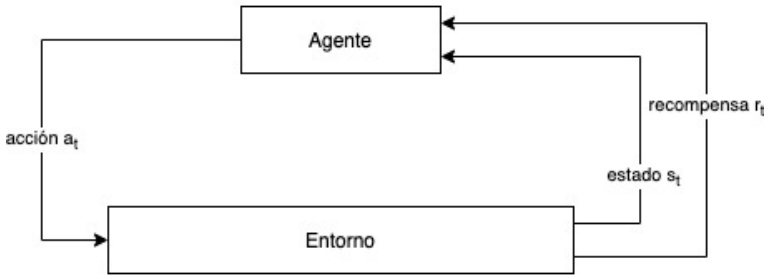

Figura 2.9: Esquema de interacción entre un agente y su ambiente.

El objetivo del agente es aprender una política $\pi : S \to A$ que retorne la próxima acción a_t basada en el estado actual s_t. De todas las políticas se debe seleccionar aquella que maximiza la recompensa. Se define la función $V^\pi(s_t)$ como el valor acumulado de recompensa obtenida partiendo del estado s_t como

$$V^\pi(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... = \sum_{i=0}^{\infty} \gamma^i r_{i+1} \tag{2.3}$$

donde $0 \leq \gamma < 1$ es una constante denominada *factor de descuento* que determina el valor relativo de las recompensas futuras contra las recompensas inmediatas. Si el valor de γ es igual a cero, solo se considera el valor de la recompensa inmediata. Cuando el valor de γ se acerca a uno, las recompensas a futuro tienen un mayor peso frente a las inmediatas.

De esta forma el agente debe aprender una política que maximice $V^\pi(s_t)$, es decir

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, V^\pi(s), (\forall s) \tag{2.4}$$

Para modelar el problema de este trabajo como una tarea de aprendizaje por refuerzo se definen los estados como la oración generada hasta un determinado momento por el generador, y a la acción como la elección de la siguiente palabra de la oración. Una vez que el generador selecciona la acción que determina el fin de oración, recibe

una recompensa otorgada por el discriminador que representa qué tan bueno fue el diálogo generado. Dado esto, se puede interpretar al discriminador como la función recompensa. Para que el agente determine la siguiente palabra es necesario contar con una política, la cual dado un estado y un parámetro θ retorna la siguiente acción.

El entrenamiento del generador consiste en encontrar una política óptima, es decir, encontrar los parámetros óptimos θ que maximicen la recompensa dada por el discriminador. Sin embargo, elegir una palabra dentro de un vocabulario determinado es una tarea discreta. Esto implica que los pesos del generador no se pueden actualizar mediante el algoritmo *backpropagation*[3] (Yu et al., 2017), por lo que se utiliza el algoritmo REINFORCE (sección 2.4.3).

2.4. Algoritmo REINFORCE

El algoritmo REINFORCE consiste en aprender una política parametrizable a través de la aproximación del método gradiente de política. Esta política retorna las probabilidades de las posibles acciones dado el estado actual. Se basa en la idea de que durante el proceso de aprendizaje las acciones que generan una buena recompensa deben ser más probables que las acciones que generan malas recompensas (Graesser & Keng, 2019). Este algoritmo tiene tres componentes: un objetivo a ser maximizado, una política parametrizable, y un método para actualizar los parámetros de la política. En las siguientes subsecciones se detalla cada uno de ellos.

2.4.1. Función Objetivo

Cuando un agente interactúa con su entorno genera una trayectoria que contiene una secuencia de recompensas. La trayectoria se define como

$$\tau = \langle s_0, a_0, r_0 \rangle, ..., \langle s_t, a_t, r_t \rangle \tag{2.5}$$

y el valor acumulado de la trayectoria como

$$V(\tau) = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} \tag{2.6}$$

El *objetivo* es el retorno esperado sobre todas las trayectorias completadas por el agente y se puede definir como

$$J(\pi_\theta) = \mathbb{E}_{t\sim\pi_\theta}\left[V(\tau)\right] = \mathbb{E}_{t\sim\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right] \tag{2.7}$$

La ecuación anterior se puede interpretar de manera que el valor esperado es calculado sobre varias trayectorias generadas a partir de una política, esto es $\tau \sim \pi_\theta$.

2.4.2. Gradiente de política (*Policy Gradient*)

La política π_θ es una función que permite a un agente interactuar con un ambiente, y el objetivo provee una meta a maximizar. El algoritmo gradiente de política resuelve

[3] *Backpropagation* es un algoritmo utilizado para actualizar los pesos de una red neuronal (Mitchell, 1997).

el problema de encontrar los parámetros θ de la política tal que el objetivo sea máximo, es decir

$$\max_{\theta} J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[V(\tau)\right] \tag{2.8}$$

Para maximizar el objetivo se usa el método ascenso por gradiente sobre los parámetros de la política (el gradiente apunta a la dirección de mayor ascenso). Los parámetros θ se actualizan de la forma

$$\theta = \theta + \alpha\nabla_\theta J(\pi_\theta) \tag{2.9}$$

donde α es una constante denominada *factor de aprendizaje* que se utiliza para regular la velocidad de actualización del parámetro. El término $\nabla_\theta J(\pi_\theta)$ se conoce como *gradiente de política* y se define como

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} V_t(\tau)\nabla_\theta \log \pi_\theta(a_t|s_t)\right] \tag{2.10}$$

donde $\pi_\theta(a_t|s_t)$ es la probabilidad de tomar una acción en tiempo t.

Por lo tanto, el método gradiente de política es un mecanismo que genera cambios en las probabilidades de las acciones tomadas por la política. Si $V_t(\tau) > 0$ la probabilidad de la acción $\pi_\theta(a_t|s_t)$ es incrementada, mientras que si $V_t(\tau) < 0$ la probabilidad disminuye. Luego de varias actualizaciones del parámetro θ utilizando la ecuación 2.9, la política aprende a producir acciones que maximicen $V_t(\tau)$

2.4.3. Detalle del algoritmo REINFORCE

En el algoritmo 1 se presenta el pseudocódigo de REINFORCE. Primero se inicializa el factor de aprendizaje α y se construye una política π_θ inicializando los pesos de la red con valores aleatorios. Luego se itera sobre múltiples episodios y en cada uno se usa π_θ para generar una trayectoria $\tau =< s_0, a_0, r_0 >, ..., < s_t, a_t, r_t >$. Luego, en cada paso t de la trayectoria, se calcula el retorno $R_t(\tau)$ para estimar el gradiente de política, que se acumula con el fin de actualizar los parámetros θ de la política.

Algorithm 1 Pseudoódigo del algoritmo REINFORCE.

Inicializacion de variable de aprendizaje α
Inicializacion de pesos θ para la politica π_θ.
for episodio = 0,...,MAX_EPISODIOS **do**
  obtener una trayectoria $\tau =< s_0, a_0, r_0 >, ..., < s_T, a_T, r_T >$
  set $\nabla_\theta J(\pi_\theta) = 0$
  for t=0,...,T **do**
    $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r'_t$
    $\nabla_\theta J(\pi_\theta) = \nabla_\theta J(\pi_\theta) + R_t(\tau)\nabla_\theta log\pi_\theta(a_t|s_t)$
  $\theta = \theta + \alpha\nabla_\theta J(\pi_\theta)$

El algoritmo REINFORCE utiliza el método Monte Carlo para estimar numéricamente el gradiente de política. Este método se refiere a cualquier mecanismo que haga uso del muestreo aleatorio para generar datos que luego son utilizados para aproximar una función. Es así que de forma sencilla se puede aproximar la función del gradiente de política. El valor esperado $\mathbb{E}_{\tau\sim\pi_\theta}$ implica que mientras más trayectorias τ son

muestreadas usando la política π_θ, y al realizar el promedio, más se aproxima al valor real de $\nabla_\theta J(\pi_\theta)$. El gradiente de política es implementado como una búsqueda Monte Carlo estimada sobre las trayectorias (Graesser & Keng, 2019).

Al utilizar el método de búsqueda Monte Carlo, el gradiente de política estimada puede tener una alta varianza dado que el retorno puede variar de una trayectoria a otra. Debido a que las acciones son aleatorias al ser muestreadas a partir de una distribución de probabilidad, el estado inicial puede variar entre episodios y la función de transición puede ser estocástica. Entonces una forma de reducir la varianza es modificar el retorno de tal forma que se le reste un valor ($b(s_t)$) que no dependa de la acción, es decir

$$\nabla_\theta J(\pi_\theta) \approx \sum_{t=0}^{T} (V_t(\tau) - b(s_t)) \nabla_\theta \log \pi_\theta(a_t|s_t) \tag{2.11}$$

2.5. *Reward for every generation step* (REGS)

El método básico de REINFORCE (sección 2.4) provoca que la recompensa generada por el discriminador sea a la oración completa y no paso a paso. Es decir, el generador no tiene forma de saber si parte de la oración generada fue buena o mala. Por ejemplo, supongamos que el histórico de entrada es *«¿Cuál es tu nombre?»* y la respuesta humana es *«Yo soy Pedro»*, mientras que la respuesta por el modelo es *«Yo no lo sé»*. El algoritmo básico asociará la misma recompensa negativa a todos los *tokens* generados. Pero la asignación de recompensa adecuada debería otorgar una recompensa neutral al *token «Yo»* y una recompensa negativa a los *tokens «no»*, *«lo»* y *«sé»*. Este proceso de retorno parcial de la recompensa se denomina REGS (Li et al., 2017).

Esta técnica consiste en utilizar el método Monte Carlo para generar subsecuencias de una oración. Cada subsecuencia es procesada por el discriminador y de esta forma el generador se alimenta con la recompensa de las secuencias parciales. Dada una subsecuencia *sp* el modelo va a completar la oración N veces. De esta forma las N oraciones generadas van a compartir el prefijo *sp* y cada una de ellas va a ser enviada al discriminador. El puntaje promedio se utiliza como recompensa del prefijo *sp*.

La desventaja de este método es que hay que repetir el proceso de generación N veces y esto puede ser computacionalmente costoso.

2.6. *Teaching Forcing*

En la arquitectura GAN básica sucede que el generador solamente es capaz de generar las respuestas óptimas a través de la recompensa del discriminador. Esto provoca que se genere un entrenamiento inestable tanto con el uso de REINFORCE como con REGS. Cuando en repetidas ocasiones el generador retorna sentencias que son clasificadas como provenientes del algoritmo y no generadas por un humano, este deteriora cada vez más su rendimiento y no puede mejorar la calidad. A través de la recompensa el generador sabe que generó una mala oración, pero no sabe cuál era la correcta ni cómo generarla. Para sortear este problema Li et al. (2017) proponen el uso de *Teaching Forcing*.

Este método consiste en forzar la salida del generador con oraciones generadas por humanos para que el discriminador le asigne una recompensa con valor uno (u otro

valor positivo) a estas oraciones, y este sea utilizado por el generador para actualizar su modelo. Se puede pensar en esta estrategia como un profesor que regula al modelo una vez que este empieza a desviarse del conjunto de entrenamiento.

En este capítulo se presentaron los métodos y conceptos teóricos utilizados por los modelos de este trabajo. En el capítulo siguiente se presenta en detalle la arquitectura GAN, la estructura del generador y el discriminador, y el proceso de entrenamiento de cada uno de ellos junto con el proceso de construcción del corpus utilizado por este.

Capítulo 3

Generación de diálogo utilizando RL y GAN

Los objetivos originales de este proyecto son: generar diálogos dentro de un dominio abierto en idioma inglés y español, y utilizar diferentes modelos para realizar una comparativa entre ellos. Luego de la investigación inicial se decide acotar el dominio a diálogos generados por personas de diferente género. Durante la investigación de los trabajos sobre este tema se analizan los corpus utilizados por distintos autores y se observa que los corpus en español presentan dificultades en su uso debido al formato y su contenido. Es por esto que se decide utilizar solamente diálogos en idioma inglés.

Por otra parte, se estudian los modelos de los trabajos propuesto por Li et al. (2018) (DG-AIRL y DG-AIL) y Li et al. (2017)(GAN+REINFORCE), con el objetivo de realizar una comparativa entre ellos. A la hora de entrenar los modelos DG-AIRL y DG-AIL se presentan dificultades que no se logran sortear. Por lo que finalmente se decide utilizar y modificar la implementación propuesta por Li et al. (2017) para realizar una evaluación de las respuestas generadas e interactuar con el modelo.

En la sección 3.1 se describen los corpus analizados, incluyendo el corpus seleccionado, y en la sección 3.2 la arquitectura del modelo. Por otro lado, en la sección 3.3 se presenta el ambiente y proceso de entrenamiento, y en la última (sección 3.4) se detalla la implementación de un chat web para interactuar con los modelos.

3.1. Corpus

Los corpus que se analizaron en la investigación inicial fueron *Ubuntu Dialogue corpus*[1], *OpenSubtitles corpus*[2] y *Reddit All Comments Corpus*[3]. El primero de ellos está constituido por registros de chats sobre temas relacionados a Ubuntu. Si bien este corpus está relacionado a un tema en particular, contiene una gran cantidad de diálogos de varios turnos entre dos participantes. *Reddit All Comments Corpus* está constituido por conversaciones de subforos de la plataforma *Reddit*. Los aspectos a destacar de este corpus es que contiene conversaciones de temas variados y su tamaño (1.7 mil millones de comentarios). El punto en contra de este corpus es que *Reddit* utiliza hilos para los comentarios, por lo que en los diálogos pueden participar más de dos personas. El corpus *OpenSubtitles corpus* contiene los subtítulos de una gran cantidad y variedad de películas, y las traducciones entre distintos idiomas para un mismo subtítulo. Los archivos contienen diálogos planos y no contienen información de género sobre los personajes. En el cuadro 3.1 se presenta en detalle las características de los corpus mencionados.

[1] `https://github.com/rkadlec/ubuntu-ranking-dataset-creator`

[2] `http://opus.nlpl.eu/OpenSubtitles-v2018.php`

[3] `https://www.reddit.com/r/datasets/comments/3bxlg7/i_have_every_publicly_available_reddit_comment/`

Corpus	# de oraciones	# de diálogos	# de palabras
Ubuntu Dialogue corpus	-	930,000	100,000,000
Open Subtitles corpus	140,000,000	36,000,000	1,000,000,000
Reddit All Comments Corpus	3,329,219,008	-	-
Cornell Movie-Dialogue	305,000	220,000	9,000,000

Cuadro 3.1: Detalle de los corpus analizados.

Finalmente, el corpus seleccionado fue *Cornell Movie Dialogs Corpus*, que contiene una colección de diálogos de películas americanas y sus metadatos. La razón por la que se selecciona este corpus es porque entre estos metadatos se encuentran los géneros de las películas, qué personajes participan de los diálogos e información sobre cada personaje. Un punto a observar es que este corpus, al estar formado por diálogos de películas, contiene un cierto grado de oralidad. El uso del corpus para este proyecto se basa en los archivos que se presentan en el cuadro 3.2.

Archivo	Contenido	Formato
`movie_lines.txt`	Archivo que contiene las líneas de las conversaciones	`ID linea $ ID Pers. $ ID Película $ Nombre Pers. $ Texto del enunciado`
`movie_ conversations.txt`	Archivo que contiene los metadatos de las conversaciones de las películas	`ID Primer Pers. $ ID Segunda Pers. $ ID Película $ Lista de enunciados de la conv.`
`movie_characters_ metadata.txt`	Archivo que contiene los metadatos de los personajes que participan en las películas	`ID Pers. $ Nombre Pers. $ ID Película $ Título Película $ Género $ Posición en los créditos`
`movies_titles_ metadata.txt`	Archivo que contiene los metadatos de las películas	`ID Película $ Nombre Película $ Año Película $ Puntaje IMDB $ Cant. IMDB votos $ Lista de géneros de la película`

Cuadro 3.2: Descripción de los archivos utilizados.

En el cuadro 3.3 se presentan ejemplos de los archivos con la información de cada película. Como se puede ver hay identificadores que permiten indexar los archivos entre sí. Por ejemplo, el archivo `movie_titles_metadata.txt`, contiene el identificador `m0` que representa la película *10 things I hate about you*. Este identificador es utilizado para recuperar los diálogos y personajes que conforman a la película (`u0`, `u5`, `L542` y `L543` respectivamente en el ejemplo del archivo `movie_conversation.txt`). Luego con los identificadores del diálogo y del personaje se obtiene el contenido del diálogo (archivo `movie_lines.txt`) y las características de los personajes del archivo `movie_characters_metadata.txt` (en particular el género).

Archivo	Ejemplo
movie_titles_metadata.txt	`m0 $ 10 things i hate about you $ 1999 $ 6.90 $ 62847 $ ['comedy', 'romance']`
movie_conversation.txt	`u0 $ u5 $ m0 $ ['L542', 'L543']`
movie_lines.txt	`L542 $ u0 $ m0 $ BIANCA $ You are so completely unbalanced.`
movie_characters_metadata.txt	`u0 $ BIANCA $ m0 $ 10 things i hate about you $ f $ 4`

Cuadro 3.3: Ejemplos de los archivos que componen el corpus *Cornell Movie Dialogs*.

Como se puede observar en el cuadro 3.3, utilizando estos archivos se pueden generar todas las conversaciones de cada película. Para este trabajo se necesita solo algunas de ellas y se deben estructurar para que sean compatibles con los modelos a estudiar.

En este proyecto se opta por trabajar con diálogos románticos porque es más probable que ocurran diálogos entre personas de distinto género. Lo primero que se debe hacer es filtrar las películas para el entrenamiento. Para esto simplemente se seleccionan las películas que contengan la palabra `romance` dentro de la lista de géneros en el archivo que contiene los metadatos de las películas. Luego, se filtra el archivo que contiene las conversaciones y el archivo que contiene los metadatos de los personajes utilizando los identificadores de las películas obtenidos del paso anterior.

Al terminar estos pasos debemos obtener el texto de la conversación. Para poder realizar esto, lo primero que se debe chequear es que los personajes que pertenecen a la conversación sean de géneros distintos. Hay que tener cuidado, ya que los creadores del corpus no siempre pudieron determinar el género de los personajes, por lo que en ciertos casos se encuentra el símbolo `?` en el género del personaje. Usando el archivo de los metadatos de los personajes y el archivo de las conversaciones, se puede cruzar qué personajes tienen los valores correctos de género y pertenecen a las conversaciones filtradas previamente. Finalmente, se pueden generar los diálogos a partir de los identificadores de las oraciones anteriores y el archivo que contiene cada oración.

Al mismo tiempo que se obtienen los textos de las conversaciones, se comienza a construir la estructura necesaria para hacer uso del modelo. El modelo hace uso de un conjunto de preguntas (`.query`) y un conjunto de respuestas (`.answer`) que son los enunciados que forman parte de la conversación. Para este trabajo se construyen dos conjuntos de datos, ya que el objetivo es poder predecir la respuesta de el género masculino y femenino, por lo que los resultados de este proceso son:

- `woman.query` y `man.answer`, los cuales se utilizan para entrenar el modelo que predice la respuesta de genero masculino
- `man.query` y `woman.answer`, los cuales se utilizan para entrenar el modelo que predice la respuesta de genero femenino

Los enunciados que forman parte de los conjuntos previos deben ser sometidos a distintas modificaciones ya que el modelo necesita cierto formato de *tokens*. Las modificaciones consisten en reemplazar signos de puntuación, agregar distintos *padding* para ciertos símbolos y remover otros caracteres [4].

[4]Sustituir guiones bajos por apóstrofos, agregar un espacio previo a un apóstrofo, signos de puntuación y signos de porcentajes; agregar espacios separadores dentro de paréntesis y comillas dobles, agregar espacio posterior al símbolo de moneda, agregar un espacio previo y luego de arroba y *ampersand*, convertir letras mayúsculas a minúsculas y remover *pipes* (símbolo "|").

En el cuadro 3.4 se presenta un ejemplo de un enunciado antes y después del procesamiento.

Previo del procesamiento	`Who knows? All I've ever heard her say is that she'd dip before dating a guy that smokes.`
Luego del procesamiento	`who knows ? all i 've ever heard her say is that she 'd dip before dating a guy that smokes .`

Cuadro 3.4: Ejemplo de un enunciado antes y después de realizar el procesamiento.

Como se dijo anteriormente, el *script* genera dos conjuntos de datos para poder predecir diálogos tanto de género masculino como femenino. Se puede ver en el cuadro 3.6 que la cantidad de diálogos comenzados por mujeres es mayor al comenzado por hombres, por lo que se removieron los últimos diálogos del archivo de diálogos comenzados por mujeres para que ambos tengan la misma cantidad. Esto fue necesario para poder realizar el entrenamiento y que ambos entrenen con el mismo conjunto de datos. En el cuadro 3.5 se muestra un ejemplo de un diálogo con actores de distinto género.

Actor 1 (Mujer): *why can 't we agree on this ?*
Actor 2 (Hombre): *because you 're making decisions for me .*

Cuadro 3.5: Ejemplo de un diálogo entre actores de diferente género.

# de películas románticas	132
# de conversaciones románticas	20158
# de pares de diálogos románticos	57360
# de enunciados románticos	77518
# de conversaciones románticos entre personajes de distinto género	7904
# de pares de diálogos románticos entre personajes de distinto género	24298
# de pares de diálogos románticos comenzados por el género masculino	11902
# de pares de diálogos románticos comenzados por el género femenino	12396

Cuadro 3.6: Resultados del *script* sanitizador.

3.2. Arquitectura

El modelo debe generar una respuesta $y = y_1, y_2, ..., y_T$ dado un historial x, el cual como mencionan los autores de los modelos utilizados en este trabajo (Li et al., 2017), está formado por una secuencia de dos oraciones ya que utilizar más de dos no presenta mejoras significativas en el rendimiento. El proceso de generar la respuesta y es visto como una secuencia de acciones (donde cada acción consiste en seleccionar un *token*) que son seleccionadas conforme a una política definida por una red neuronal *encoder-decoder* (Li et al., 2017). Para ello se utiliza una arquitectura GAN que consiste en dos modelos: un generador (G) y un discriminador (D). La tarea del generador es construir oraciones tales que el discriminador no pueda detectar que fueron generadas por él. En la figura 3.1 se presenta un esquema de la arquitectura.

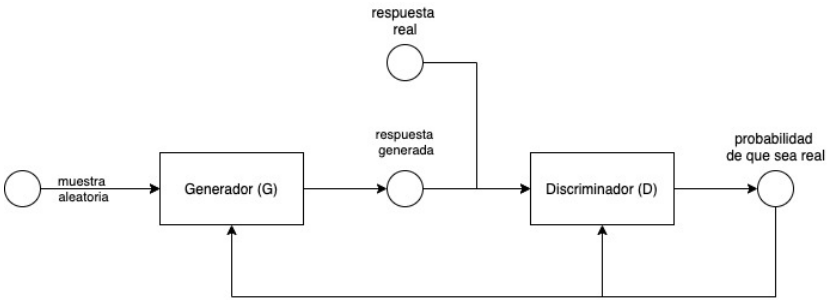

Figura 3.1: Estructura del modelo GAN.

G implementa un modelo Seq2Seq (sección 2.1.4) para generar una respuesta y a un diálogo previo x. Para ello se utilizan dos LSTM: una en el *encoder* y otra en el *decoder*. El *encoder* hace uso de una capa de *embeddings* para transformar el diálogo x en una lista de vectores, donde cada vector es la representación de una palabra del diálogo, y es enviado de a uno a la red LSTM. En cada paso recibe como entrada el vector de estado generado hasta el momento y el *embedding* de la siguiente palabra del diálogo. Una vez que se envía el último token (`<EOS>`) se utilizan los valores de la última capa oculta de la red para formar el vector de representación de tamaño fijo de x (vector contexto). Este vector, junto con el token `<BOS>`, es utilizado como entrada en el paso inicial del *decoder*. Al igual que el *encoder*, en cada paso recibe el vector de estado, y en lugar de recibir el *embedding* de la siguiente palabra, recibe el de la palabra generada en el paso anterior. Para construir la oración y, en cada paso se utiliza una capa *softmax* para obtener las probabilidades de selección de la siguiente palabra (la palabra seleccionada es aquella con mayor probabilidad). En la figura 3.2 se presenta la arquitectura de este modelo.

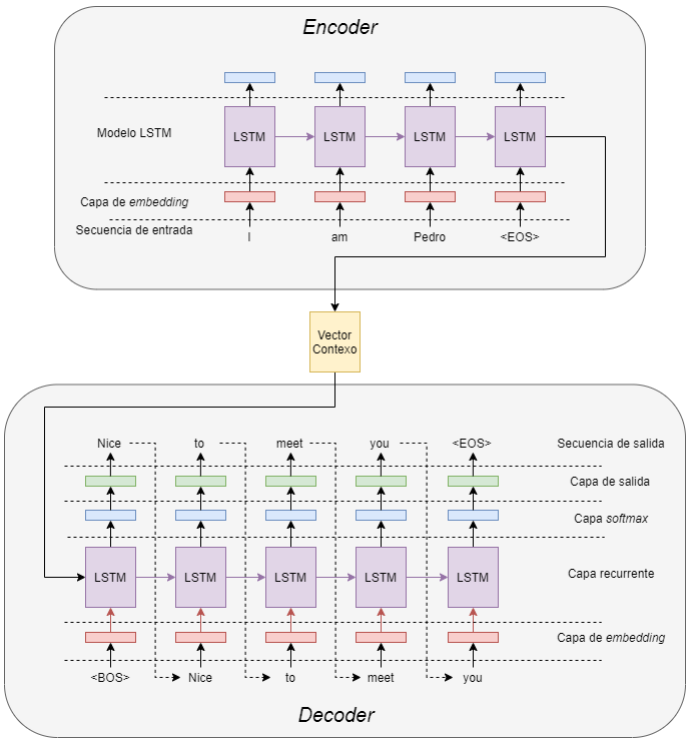

Figura 3.2: Arquitectura del modelo Seq2Seq utilizado en el generador.

El objetivo de G es representar la política parametrizada $\pi(a|s, \theta)$, donde θ corresponde a los parámetros de la red que se actualizan mediante el algoritmo REINFORCE (sección 2.4). Por lo tanto, el generador debe recibir como entrada una secuencia de palabras y generar una distribución de probabilidad con la palabra siguiente. Para lograr que el generador reciba recompensas parciales, y no una única al final de cada oración, se utiliza el método Monte Carlo. La idea es comenzar a partir de un estado s y simular N episodios[5] diferentes siguiendo la política π. Esto significa que G genera N oraciones diferentes con el mismo prefijo s. Por ejemplo, dado el prefijo s = *«El perro»* el generador podría producir oraciones como: *«El perro está en la casa del vecino»*, *«El perro corrió detrás de un auto»*, *«El perro tiene siete meses»*, etc. Luego se obtiene la recompensa correspondiente a cada una de las N oraciones usando la salida del discriminador y se promedian para obtener la recompensa parcial asociada al estado s. La figura 3.3 ilustra lo explicado anteriormente.

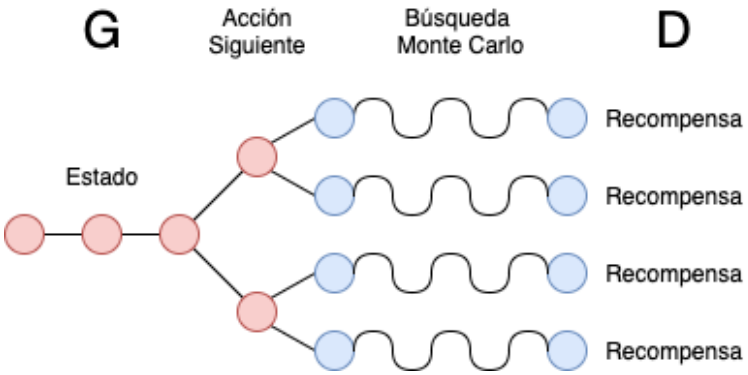

Figura 3.3: Ilustración de la utilización del método Monte Carlo para obtener recompensas parciales.

D es un clasificador binario que recibe una entrada de la forma $\{x, y\}$ y retorna la probabilidad de que sea generada por un humano o por una máquina, la que es utilizada como recompensa para entrenar a G. La entrada del discriminador es transformada a una representación vectorial utilizando un *encoder* jerárquico, que luego es utilizada como entrada a una capa *2-class-softmax* que retorna la probabilidad de que haya sido generada por una máquina o por un humano.

La representación jerárquica fue propuesta por Li et al. (2015) y está inspirada en la yuxtaposición de palabras para crear oraciones y yuxtaposición de oraciones para crear un párrafo. Este esquema consiste en dos niveles de *encoders* análogos a los utilizados en el generador. El primer nivel es utilizado para obtener la representación vectorial del diálogo previo x y la respuesta generada y, y el segundo nivel es utilizado para, a partir de estos estos vectores, generar una representación vectorial única del diálogo completo. Finalmente, esta representación es la utilizada como entrada a la capa *softmax*. En la figura 3.4 se presenta un esquema de esta estructura.

[5]Li et al. (2017) proponen utilizar $N = 5$.

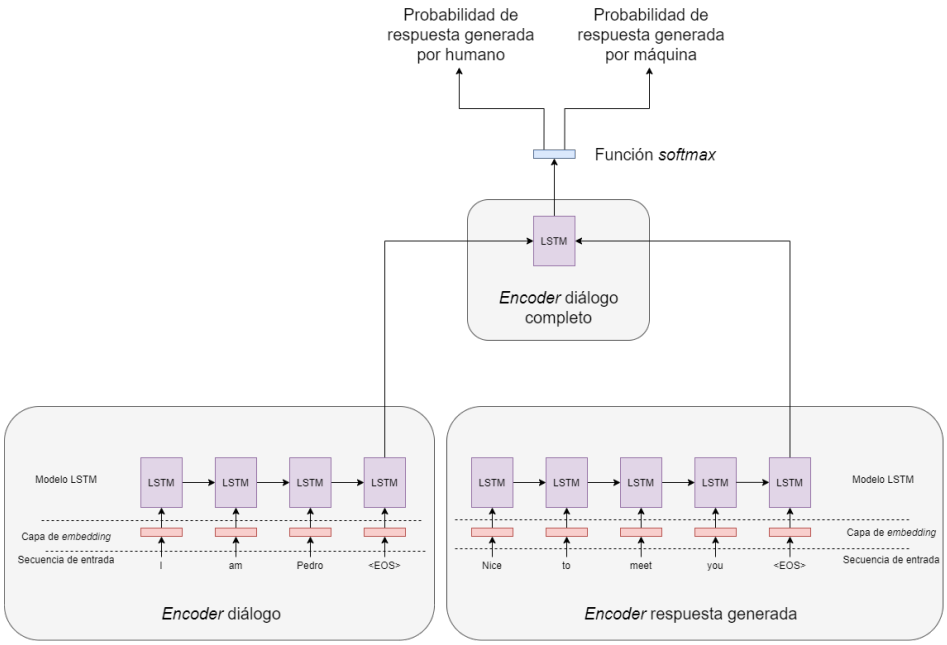

Figura 3.4: Arquitectura del modelo discriminador.

Algunas particularidades de la implementación son: se remueven las oraciones del conjunto de entrenamiento cuya cantidad de palabras es menor a 5, y se penalizan las palabras que ya fueron generadas para prevenir que haya repetidas. Por otro lado se utiliza la técnica *Teaching Forcing* en el entrenamiento para evitar que se degrade el rendimiento cuando el generador deteriora la calidad de las respuestas generadas. Para ello se alimenta al generador con ejemplos extraídos del corpus para que el discriminador retorne una recompensa positiva que será usada por el generador para actualizarse. En el algoritmo 2 se presenta el pseudocódigo del algoritmo utilizado.

Algorithm 2 Pseudocódigo del algoritmo utilizado.

for i=1...N=número de iteraciones **do**
  for j=1..D=número de pasos para el discriminador **do**
    obtener el par (X, Y) del corpus
    generar $Y^{'} = G(\ldots|X)$
    actualizar D usando (X, Y) como valor positivo y $(X, Y^{'})$ como valor negativo
  for j=1..G=numero de pasos para el generador **do**
    obtener el par (X, Y) del corpus
    generar $Y^{'} = G(\ldots|X)$
    obtener la recompensa r para $(X, Y^{'})$ usando D
    actualizar G en $(X, Y^{'})$ usando la recompensa r
    Teacher-Forcing: actualizar G en (X, Y)

3.3. Entrenamiento

En esta sección se presenta el proceso y ambiente de entrenamiento del modelo generador y discriminador. Ambos fueron entrenados con el 90 % del corpus, utilizando como condición de parada la cantidad de iteraciones. El 10 % restante se utiliza para

evaluar los resultados obtenidos (lo que se muestra en el siguiente capítulo).

La ejecución del modelo está basada en el código del repositorio [6] vinculado con el trabajo propuesto por Li et al. (2017). Para el entrenamiento del modelo se usa una computadora con un CPU Intel Core i5-4440 3.10 GHz, memoria RAM 16GB y una tarjeta gráfica GTX 1070.

Para el entrenamiento se siguen algunas de las sugerencias planteadas por Sutskever et al. (2014) y Buduma & Locascio (2017). La primera de ellas es utilizar *Bucketing*, un método utilizado en tareas Seq2Seq que ayuda al modelo a manejar de forma eficiente oraciones de diferente largo. Para ello se selecciona cada par *encode-decode* y se coloca dentro de un *bucket* en el cual todos tendrán un tamaño similar, y luego se rellenan las oraciones con un *token* especial hasta el tamaño máximo de ese *bucket*. La figura 3.5 muestra esta idea con un ejemplo.

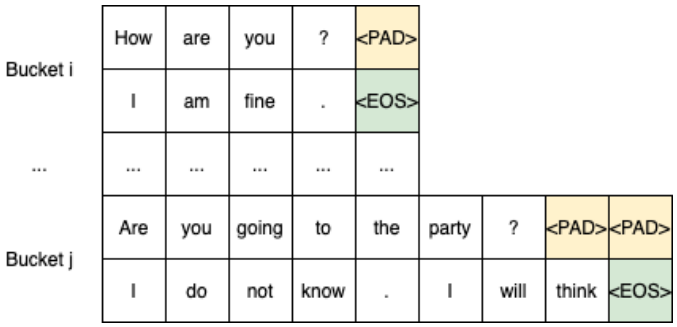

Figura 3.5: Técnica de separar en *buckets* las secuencias del conjunto de datos.

En este trabajo se utilizan cuatro *buckets*: `[(5,10),(10,15),(20,25),(40,50)]`[7]. La ventaja de utilizar esta técnica es que se aumenta considerablemente la velocidad de entrenamiento y evaluación. Es necesario añadir un nuevo *token* `<GO>` para señalar al *decoder* que debe comenzar. Los pares *encode-decode* que caen fuera de los *buckets* son descartados. Esto se debe a que oraciones muy cortas no aportan información o son demasiado genéricas, mientras que para las oraciones demasiado largas el modelo no es capaz de procesar el contexto.

Otro método sugerido es *gradient clipping*, que consiste en que si el gradiente crece demasiado se reescala para mantener en un valor más bajo y se mantenga dentro de un umbral estable. Si $||g|| \geq c$ se recalcula el gradiente de la forma: $g = c \cdot \frac{g}{||g||}$, siendo c un hiperparámetro y g el gradiente. Esto ayuda al método de descenso por gradiente a tener un comportamiento estable incluso si la pérdida del modelo es irregular. En este caso se usan los mismos valores que en el trabajo de Sutskever et al. (2014), para cada entrenamiento del *batch* se calcula $s = ||g||$ y si $s > 5$ entonces recalculamos g: $g = \frac{5g}{s}$.

La última sugerencia es decrementar el factor de aprendizaje si luego de varios entrenamientos la pérdida no cambia. Sutskever et al. (2014) plantean inicializar el factor en 0.7 y reducir a la mitad el valor cada vez que se detecta un estancamiento. Esto se hace con el objetivo de no caer en mínimos locales.

Para lograr utilizar el código base es necesario realizar algunos ajustes de configuración para lograr utilizar el corpus construido y la tarjeta gráfica con la que se cuenta. Por ejemplo, el código original utiliza dos capas ocultas y 1024 neuronas para

[6] https://github.com/liuyuemaicha/Adversarial-Learning-for-Neural-Dialogue-Generation-in-Tensorflow

[7] Los tamaños de los *buckets* son los que se utilizan en el trabajo de Li et al. (2017).

entrenar. Se tiene que reducir a 512 neuronas para poder entrenar usando la tarjeta gráfica mencionada anteriormente. Al igual que en el código original, para todos los entrenamientos se utiliza un vocabulario de 25000 palabras.

Por otro lado, en el repositorio no se especifica la condición de parada de los entrenamientos. Para analizar el comportamiento de los modelos y luego poder determinar una condición de parada, se grafican los pasos y los valores de la función de pérdida del entrenamiento del discriminador y del generador (utilizando el método de entrenamiento adversario). En las figuras 3.6 y 3.7 se puede ver como evoluciona la pérdida del discriminador a medida que avanzan los pasos, tanto para el modelo femenino como el masculino. Para ambos se ve que mientras se acerca a los 20000 pasos la pérdida se acerca a cero.

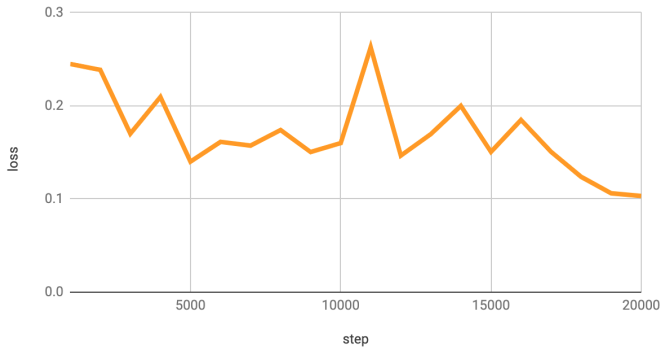

Figura 3.6: Gráfica de la pérdida en función de los pasos del discriminador para el modelo masculino.

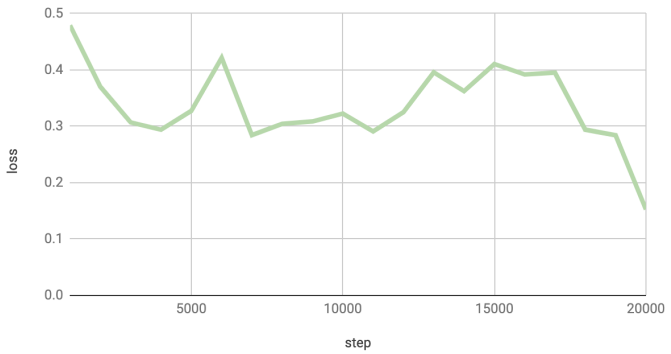

Figura 3.7: Gráfica de la pérdida en función de los pasos del discriminador para el modelo femenino.

Un fenómeno similar ocurre para el entrenamiento del generador utilizando la arquitectura GAN (figura 3.8 y 3.9). Dado estos fenómenos se decide finalizar el entrenamiento de ambos luego de la ejecución de 20000 pasos, lo que lleva aproximadamente 24 horas de ejecución en el *hardware* mencionado.

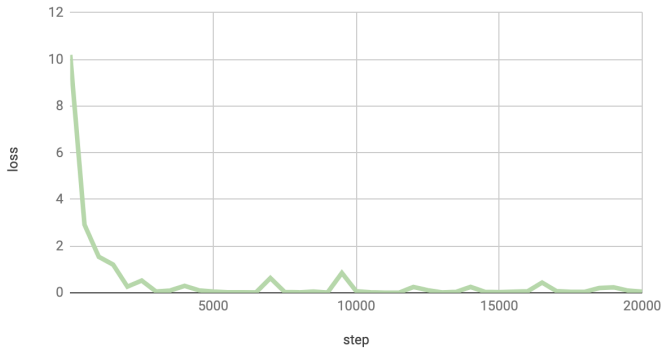

Figura 3.8: Gráfica de la pérdida en función de los pasos del generador con el entrenamiento GAN para el modelo femenino.

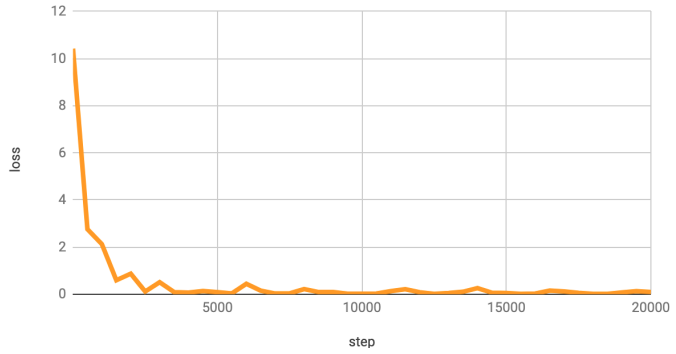

Figura 3.9: Gráfica de la pérdida en función de los pasos del generador con el entrenamiento GAN para el modelo masculino.

Un punto a notar es que primero se preentrena el modelo generador usando una red Seq2Seq utilizando el corpus construido, y luego se preentrena el discriminador utilizando como ejemplos negativos los diálogos generados por el modelo generador preentrenado. El preentrenamiento del generador no se midió porque el resultado interesante es el retornado por el entrenamiento del generador mediante la arquitectura GAN, la que es utilizada para realizar la evaluación de las respuestas generadas, y por lo tanto, la calidad final del modelo generador.

3.4. Interacción con el modelo

Para poder interactuar con los modelos se utiliza una implementación de un chat web mediante el cual, a través de dos botones, se puede seleccionar el género del modelo y comenzar una conversación. La figura 3.10 muestra la interfaz gráfica. La implementación consiste en un *frontend* y un *backend* que tiene como tarea principal levantar la instancia del modelo correspondiente y enviar al modelo el texto que ingresa el usuario por pantalla. Una vez que el modelo procesa la respuesta, es enviada al *frontend* para desplegarla en el chat. En la figura 3.11 se presenta la arquitectura de la implementación. Algo a destacar es que cada vez que el usuario escribe en el chat, se abre y cierra una sesión[8], por lo que las conversaciones son de un solo turno.

[8]Una sesión representa una instancia del modelo. Dada un GPU, no pueden ejecutar al mismo tiempo dos sesiones de un mismo modelo. Este problema no se sortea porque el objetivo de la implementación era realizar una prueba de concepto.

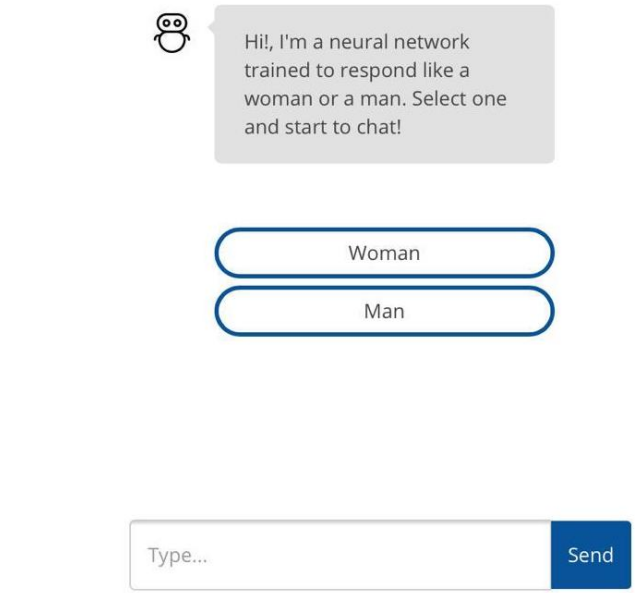

Figura 3.10: Interfaz gráfica de la implementación web del chat.

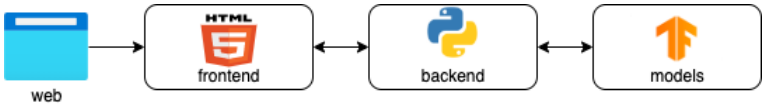

Figura 3.11: Esquema de arquitectura de la implementación del chat web.

En este capítulo se presenta cómo se implementan y entrenan los modelos propuestos por Li et al. (2017), además del proceso de creación del corpus que se usa para el entrenamiento. En el capítulo siguiente se explican los distintos métodos de evaluación y un análisis de los resultados obtenidos.

Capítulo 4

Evaluación

En este capítulo se presenta la evaluación del modelo construido y un análisis sobre los resultados obtenidos. La evaluación se realiza utilizando dos métodos: métricas basadas en heurísticas y experimentos realizados por jueces. El primero se lleva a cabo utilizando la representación vectorial de las palabras, y el segundo mediante dos formularios donde los jueces deben indicar si una respuesta fue generada por una máquina o no y determinar la calidad en base a un rango numérico.

La calidad de una conversación se puede describir en base a varios factores, como la sintaxis, la gramática y la semántica, entre otros. La tarea de evaluar la calidad de un algoritmo que genera diálogos se puede realizar con métricas sobre los enunciados, pero no dan un buen resultado y no se encontró aún una métrica que pueda evaluar un texto por sus características humanas. Es por esto que los diálogos generados deben ser evaluados por un humano para que se tenga en cuenta los factores mencionados anteriormente. En particular, nuestro modelo genera una respuesta a un enunciado sin tener en cuenta el contexto donde ocurre, por lo que determinar la calidad del modelo se deriva en evaluar la respuesta. Para ello se utilizan dos enfoques, uno a través de heurísticas y otro mediante jueces.

Con las heurísticas se busca determinar qué tan buena es una respuesta a un diálogo comparándola con la respuesta original del corpus, mientras que con la evaluación por jueces se busca cubrir aspectos "humanos" que no son posibles de calcular con las heurísticas. Las heurísticas utilizadas para la evaluación son métricas basadas en los *embeddings* de las respuestas, y la evaluación humana está inspirada en el test de Turing (Turing, 1950).

Tanto en la evaluación por jueces como en la evaluación basada en heurísticas se utilizan las mismas instancias de los modelos. Ambos fueron entrenados tal como se especifica en la sección 3.2.

En las siguientes secciones se presenta la evaluación basada en heurísticas (sección 4.1), la evaluación por jueces (sección 4.2), y por último se realiza un análisis de los resultados obtenidos de las evaluaciones (sección 4.3).

4.1. Evaluación basada en heurísticas

Las respuestas generadas por el modelo son respuestas generativas en donde el universo de posibles respuestas correctas es muy amplio. Por ejemplo, dada la pregunta *«¿Cómo salió el partido?»*, el modelo puede generar *«Fue un partido difícil, pero ganamos»*, *«Perdimos en el último minuto»* y *«No quiero hablar del partido»* como respuestas posibles. Por ello, es necesario contar con un método de evaluación que sea capaz de determinar que las tres respuestas del ejemplo anterior sean consideradas como correctas.

Métricas como BLEU (Papineni et al., 2002) y METEOR (Banerjee & Lavie, 2005) han sido utilizadas en modelos de generación de resúmenes automáticos y modelos de

traducción. La primera de ellas analiza las ocurrencias de los n-gramas que coinciden entre la respuesta candidata y la respuesta original. METEOR crea una alineación explícita entre la respuesta original y la candidata. Esta alineación está basada en pares de *tokens* iguales, sinónimos y paráfrasis, entre otros. Como se puede ver, las métricas anteriores asumen que las respuestas generadas tienen una superposición entre las respuestas originales y las candidatas. Si bien esto es más probable que ocurra en una traducción, no lo es tanto en la generación de diálogo.

Greedy Matching (Rus & Lintean, 2012), *Embeeding Average* (Wieting et al., 2016) y *Vector Extrema* (Forgues et al., 2014) son métricas que utilizan la representación vectorial (*embeddings*) de las palabras para calcular el concepto de similitud entre dos oraciones. Para generar estos vectores se pueden utilizar métodos como *bolsa de palabras*, *one hot encoding* o modelos pre-entrenados tales que reciban como entrada un texto y retorne los vectores de las palabras. En este trabajo se opta por seguir este último enfoque utilizando el modelo pre-entrenado de Google[1] con un conjunto de datos de *Google News*[2] para implementar las últimas tres métricas mencionadas. Los valores de estas métricas se encuentran entre cero y uno, mientras más alto sea el valor para una respuesta candidata, más similitud semántica presenta con la respuesta original.

Estas tres últimas métricas son elegidas ya que son las utilizadas por Li et al. (2019) para comparar sus modelos con el modelo que se basa este trabajo. Esto permite realizar una comparación entre los resultados obtenidos por los autores de Li et al. (2017) y los de este trabajo.

4.1.1. *Greedy Matching*

Este método consiste en, dada la respuesta original del corpus r y la respuesta generada por el modelo r', para cada *token* $w \in r$ se encuentre un *token* $w' \in r'$. Se encuentra w' mediante el cálculo de la similitud de los cosenos de los vectores de *embeddings* de w y w'. Luego, la medida final se calcula como el promedio de todas las palabras de la respuesta, es decir

$$G(r, r') = \frac{\sum_w \text{máx}_{w' \in r'} \, cos\_sim(e_w, e_{w'})}{|r|} \tag{4.1}$$

Como la fórmula es asimétrica se promedia la medida G en cada dirección, siendo finalmente

$$GM(r, r') = \frac{G(r, r') + G(r', r)}{2} \tag{4.2}$$

Esta métrica tiende a favorecer respuestas que contengan palabras claves y que sean semánticamente similares a la respuesta original.

4.1.2. *Embedding Average*

Este método utiliza la composición aditiva (Mitchell & Lapata, 2010), una técnica que calcula el significado de frases al promediar las representaciones vectoriales de las palabras que las forman. Esta métrica es definida como el promedio de las representaciones vectoriales de cada palabra en la respuesta r, es decir

[1] https://code.google.com/archive/p/word2vec/
[2] news.google.com

$$\bar{e}_r = \frac{\sum_{w \in r} e_w}{|\sum_{w' \in r} e_{w'}|} \tag{4.3}$$

Para comparar la oración original con la generada se calcula la similitud por coseno entre ambas.

$$EA = cos\_sim(\bar{e}_r, \bar{e}_{r'}) \tag{4.4}$$

4.1.3. *Vector Extrema*

De forma gráfica, las representaciones vectoriales de las palabras que están presentes en contextos similares, van a estar juntas en el espacio vectorial. Por lo tanto, palabras comunes se encuentran cerca del origen del espacio ya que ocurren en varios contextos; mientras que palabras que tengan más información de su contexto estarán más lejos del origen. Esta métrica prioriza las palabras que contienen más información sobre aquellas que son comunes.

La métrica de la respuesta r está compuesta por los valores más extremos entre todas las palabras de r en cada una de las dimensiones del espacio vectorial. Esto se define como

$$e_{rd} = \begin{cases} \text{máx}_{w \in r}\, e_{wd} & \text{si} \quad e_{wd} > |\,\text{mín}_{w' \in r}\, e_{w'd}| \\ \text{mín}_{w \in r}\, e_{wd} & \text{otro caso} \end{cases} \tag{4.5}$$

donde d corresponde a una de las dimensiones de los vectores y e_{wd} es el valor en la dimensión d de la representación vectorial e_w.

4.2. Evaluación por jueces

La evaluación por jueces se realiza para saber qué tan "humanas" son las respuestas generadas por el modelo. Para realizar esta evaluación se plantean dos experimentos, uno inspirado en el test de Turing y otro que consiste en la selección de un valor numérico en un rango determinado. Si bien se puede considerar como "humanas" respuestas que tengan en cuenta posibles aspectos como la semántica, la sintaxis, la aparición de palabras repetidas, y la falta de información o naturalidad en las respuestas; no se indica a los jueces que deben tener en cuenta estos aspectos explícitamente para no influir en la evaluación.

Ambos experimentos se realizan a través de formularios virtuales. Los diálogos utilizados son seleccionados de manera aleatoria del total de diálogos generados por ambos modelos (femenino y masculino). Un punto a observar es que no se obliga a completar todas las respuestas, sino que pueden completarse de forma parcial. Esto se hace para evitar que aquellos jueces que tienen dudas sobre la respuesta, voten de manera aleatoria.

Los diálogos presentes en los formularios tienen la estructura que se muestra en el cuadro 4.1. En ambos experimentos no se cuenta con el contexto del diálogo, por lo tanto, los jueces deben evaluar la respuesta generada teniendo en cuenta solamente el enunciado del primer actor. Además, en los experimentos no se menciona que todos los diálogos que se presentan fueron generados por el modelo.

Actor 1: *let me see. How about 0:00 at the gate of the club?*
Actor 2: *fine, see you then!*

Cuadro 4.1: Diálogo de ejemplo utilizado en los formularios de evaluación.

Cada formulario consiste en evaluar 12 diálogos seleccionados de un conjunto de 48 diálogos. De los 12, dos de ellos se mantienen fijos en todas las instancias de evaluación, y los diez restantes se seleccionan de forma aleatoria del conjunto de 48 diálogos. Los dos diálogos fijos se utilizan como testigos de los experimentos, que luego son utilizados en la evaluación para detectar posibles anomalías en las votaciones. A lo largo de esta sección cada uno de los 48 diálogos se representa por un índice de la forma *di* ($i \in [1, 48]$). El mapeo entre este índice y el diálogo se presenta en el anexo A.

El primer experimento está inspirado en el test de Turing, que es utilizado para probar las habilidades de una máquina de mostrar un comportamiento equivalente o indistinguible del de un humano. El test consiste en que un juez intente distinguir cuál de dos interlocutores es humano y cuál una máquina. El experimento consiste en presentar a los jueces un subconjunto de los diálogos generados por los modelos para que estos seleccionen si creen que los diálogos fueron generados por un humano o el modelo. De esta forma se busca obtener que tan bueno es el modelo generando oraciones parecidas a las que contestaría un humano.

El segundo experimento busca determinar qué tan buena es una respuesta generada por el modelo, en el que los jueces deben seleccionar un valor numérico comprendido entre uno y cinco. Uno representa una respuesta sin sentido, con errores gramaticales, palabras repetidas, etc. Por otro lado, cinco representa una respuesta buena, natural, que permite la continuidad del diálogo y tiene sentido.

4.3. Resultados

A continuación se presentan los resultados de ambas metodologías. En la metodología basada en heurísticas se realiza una comparación entre los resultados obtenidos por los modelos utilizados en el trabajo, la línea base, y el modelo implementado por Li et al. (2017). Para la evaluación humana se exponen y analizan los resultados obtenidos por los formularios utilizados para realizar los experimentos.

4.3.1. Resultado de la evaluación basadas en heurísticas

En este apartado se presentan los resultados obtenidos tanto para el modelo entrenado con el corpus de género femenino como el masculino. Para ambos se utilizaron 2421 respuestas generadas por los modelos.

En el cuadro 4.2 se muestran los valores obtenidos para las tres métricas utilizadas y para los diferentes modelos. El modelo *Basic Gen* representa el generador preentrenado sin la intervención del discriminador. Por otro lado, el modelo *Al Gen* representa el generador con el enfoque de entrenamiento adversario.

El modelo *baseline* es el algoritmo construido para utilizar como punto base de comparación. El algoritmo consiste en retornar una respuesta r (no generativa) a un diálogo d. Para obtener r se busca el diálogo d' más cercano a d y se retorna su respuesta asociada. d' se halla utilizando las representaciones vectoriales de las oraciones, calculando el promedio de cada una de ellas y retornando aquella cuyo promedio sea el más cercano a d.

Tanto el modelo básico como el entrenado utilizando la estrategia GAN presentan entre 12 % y 20 % de mejora respecto al modelo *baseline*, lo cual es una mejora considerable. Esto provee un indicador de que los modelos generan oraciones con una calidad aceptable.

Modelo	***Average***	***Greedy***	***Extrema***
Basic Gen (Li)	0.563 ± 0.003	**0.167 ± 0.001**	0.352 ± 0.002
SeqGan (Li)	**0.564 ± 0.003**	0.165 ± 0.001	**0.354 ± 0.002**
Basic Gen (Woman)	0.482 ± 0.007	0.159 ± 0.003	0.305 ± 0.005
Al Gen (Woman)	0.491 ± 0.007	0.153 ± 0.003	0.307 ± 0.005
Basic Gen (Man)	0.452 ± 0.008	0.147 ± 0.004	0.293 ± 0.006
Al Gen (Man)	0.458 ± 0.008	0.146 ± 0.004	0.294 ± 0.006
Baseline	0.338 ± 0.008	0.107 ± 0.003	0.119 ± 0.005

Cuadro 4.2: Rendimiento en términos de métricas de *embeddings* con intervalos de confianza del 95 %.

En el cuadro 4.2 también se muestran los resultados que obtuvieron Li et al. (2019) luego de ejecutar las mismas métricas sobre los mismos tipos de modelos (Basic Gen y el modelo GAN). Si bien los modelos utilizados en este trabajo tienen una diferencia en la configuración con respecto a la de los autores (cantidad de neuronas en las capas ocultas), además de utilizar un *hardware* y corpus diferente, se consideran estos valores como un buen punto de comparación para los modelos usados en este trabajo.

4.3.2. Resultados de la evaluación por jueces

El formulario de Turing (*f1*) es completado por 54 jueces y se obtuvo 646 respuestas, mientras que el test de escala (*f2*) fue completado por 44 jueces y se recolectaron 518 respuestas. En el anexo B se presenta la cantidad de votos que obtuvo cada diálogo.

El primer análisis que se realiza es que los jueces categorizan al 41,2 % de los diálogos como generados por una máquina, mientras que el 58,8 % de los diálogos se categoriza como generados por un humano (figura 4.1a) . Como este resultado está condicionado a la cantidad de veces que se presenta un diálogo en las instancias de los formularios se considera que un diálogo fue votado con cierta etiqueta (*máquina* o *humano*) si la mayoría de los jueces lo votaron como tal. En la figura 4.1b se puede ver el resultado del último análisis y se nota que se invierten los porcentajes de los valores de la escala.

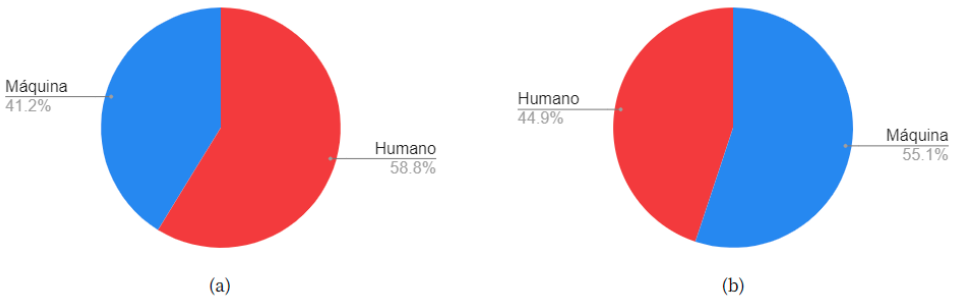

Figura 4.1: Porcentaje promedio de diálogos *máquina* vs *humano*. La gráfica (a) corresponde al porcentaje de votos sin ningún procesamiento, mientras que la gráfica (b) es la votación de los diálogos condicionado a la cantidad de veces que se presentan en el experimento.

En la figura 4.2 se exhiben los porcentajes de respuestas para cada diálogo. Se pueden observar diálogos en los que claramente la respuesta generada por el modelo no fue buena, por ejemplo, el diálogo *d1*:

Actor 1: *simple, it's fun.*
Actor 2: *let me see if you like it. You can hear it for fun in front of fun.*

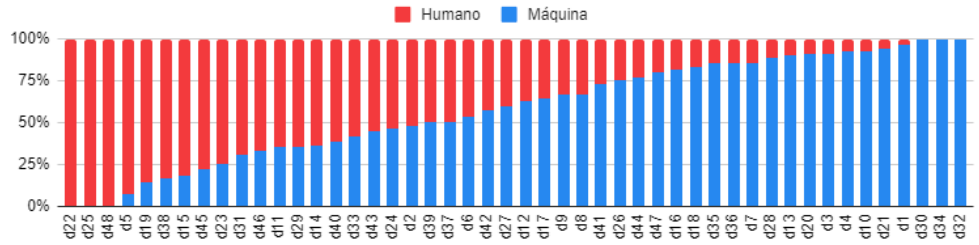

Figura 4.2: Porcentaje de votos *maquina* vs *humano* por diálogo.

Mientras que otros fueron muy buenos, por ejemplo, el diálogo *d48*:

Actor 1: *let me see. How about 0:00 at the gate of the club?*
Actor 2: *fine, see you then!*

Algunos casos interesantes son aquellos en los que los jueces tuvieron dificultades para discernir si la respuesta fue generada o no por una máquina. Estos se pueden detectar viendo que aproximadamente la mitad de los votos van para la clase *humano* y la otra mitad para la clase *máquina*. Esto puede deberse a que los jueces no contaron con el contexto completo del diálogo, por lo que dificulta la tarea de evaluar la respuesta. También hay casos en los que las respuestas son muy cortas o demasiado genéricas, lo que podría significar que fue generada por una máquina o no, como por ejemplo, el diálogo *d46*:

Actor 1: *he's not in at the moment. Can I take a message?*
Actor 2: *sure.*

En el diálogo anterior la respuesta puede ser una afirmación a la pregunta, pero al no tener el contexto completo no es posible determinar si es necesario que además de la afirmación, se agregue contenido para dar pie a que continúe el diálogo.

Si bien el experimento anterior permite realizar un primer análisis de la calidad de las respuestas generadas por el modelo, el experimento de selección de rango nos permite realizar una evaluación más detallada sobre las respuestas generadas.

El experimento de escala *(f2)* fue completado por 44 jueces y se recolectaron 518 respuestas. En este experimento se consideran tres tipos de calidad: mala (clase uno y dos), neutra (clase 3) y buena (clase 4 y 5). En la figura 4.3 se presentan los porcentajes promedio de cada clase. Se puede ver que el porcentaje de votación de las clases de calidad neutra y buena superan al porcentaje de calidad mala; mientras que en el experimento del formulario *f1* se ve lo contrario, es decir, predominan las respuestas que indican que el diálogo fue generado por el modelo. Esto puede deberse a que la cantidad de respuestas recopiladas para el segundo experimento fueron menores que las recopiladas para el primero.

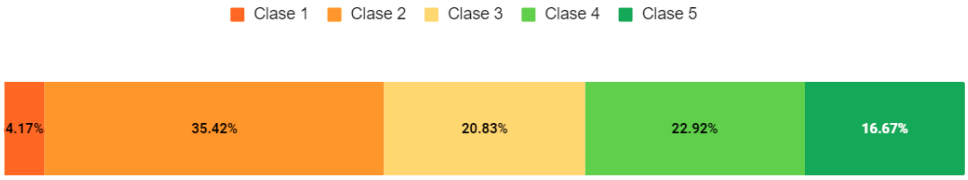

Figura 4.3: Porcentaje promedio de votos por clase.

El cuadro[3] 4.3 muestra ejemplos para los cuales es difícil identificar si el diálogo es generado por una máquina o no y evaluar si es de buena calidad. Para seleccionar estos diálogos se filtran los resultados de ambos experimentos de la siguiente manera: para el test de Turing se seleccionan aquellos diálogos cuyo promedio es $50 \pm 5\,\%$, mientras que para el experimento de escala se toman los diálogos que al sumar la cantidad de votos de las clases buenas y la cantidad de votos de las clases malas, los resultados son similares.

Diálogo	**Turing (%)**		**Escala (%)**				
	Humano	**Máquina**	**C 1**	**C 2**	**C 3**	**C 4**	**C 5**
d2	51.9	48.1	16.3	16.3	14.0	32.5	20.9
d39	50.0	50.0	18.2	18.2	9.0	36.4	18.2

Cuadro 4.3: Diálogos que tuvieron una votación pareja.

Por otro lado, en el cuadro 4.4 se presentan los diálogos que en el test de Turing tuvieron una votación similar, como se explicó previamente, pero en el análisis del experimento de escala se puede ver que tuvieron una votación tendiendo a una calidad baja o media alta (clase neutra y clases buenas). En el cuadro 4.5 se ve lo opuesto, es decir, diálogos que tuvieron una votación similar en el experimento de escala y una tendencia a que hayan sido generados por máquina o humano.

Diálogo	**Turing (%)**		**Escala (%)**				
	Humano	**Máquina**	**C 1**	**C 2**	**C 3**	**C 4**	**C 5**
d6	46.7	53.3	27.3	9.1	45.4	9.1	9.1
d24	53.3	46.7	0.0	16.7	16.7	66.6	0.0
d37	50.0	50.0	0.0	0.0	25.0	25.0	50.0

Cuadro 4.4: Diálogos que tuvieron una votación similar en el experimento de Turing pero no en el experimento de escala.

Diálogo	**Turing (%)**		**Escala (%)**				
	Humano	**Máquina**	**C 1**	**C 2**	**C 3**	**C 4**	**C 5**
d26	20.0	80.0	0.0	50.0	25.0	12.5	12.5
d28	25.0	75.0	44.5	11.1	11.1	33.3	0.0
d44	9.1	90.9	30.8	23.1	38.4	7.7	0.0

Cuadro 4.5: Diálogos que tuvieron una votación pareja en el experimento de escala pero no en el experimento de Turing.

Dos de los diálogos presentados en los formularios están en todas las instancias

[3]En los cuadros que muestran los experimento de escala, la clase i se representa con el encabezado $C\ i$.

de votación (*d1* y *d2*). Esto permite detectar si existen jueces que votan diferente a la mayoría. Se considera que un juez es distinto a la mayoría cuando selecciona una categoría que es votada pocas veces considerando la cantidad total de votos del diálogo.

El formulario *f1* fue completado por 54 jueces. El diálogo *d1* fue votado como generado por el modelo por 52 de los jueces, mientras que el diálogo *d2* tuvo una votación más equitativa: 26 de los jueces lo categorizaron como *máquina* y 28 lo categorizaron como *humano*. En el formulario *f2* se da un fenómeno similar, el diálogo *d1* fue categorizado mayormente con la clase uno, dos y tres (10, 23 y 8 votos respectivamente), mientras que el diálogo *d2* tuvo una votación más distribuida entre las clases (cuadro 4.6).

Diálogo	**Votos C 1**	**Votos C 2**	**Votos C 3**	**Votos C 4**	**Votos C 5**
d1	10	23	8	2	1
d2	7	7	6	14	9

Cuadro 4.6: Cantidad de votos por clase para los diálogos *d1* y *d2*.

Los jueces que votaron al diálogo *d1* como generado por un humano fueron *anónimo 20* y *anónimo 24*, mientras que los jueces que votaron por la clase cuatro y cinco fueron *anónimo 11*, *anónimo 19* y *anónimo 23*. Dado los resultados (cuadro 4.7) que se obtienen al no considerar estos jueces, se concluye que no son anómalos.

Jueces	**Turing (%)**		**Escala (%)**				
	Humano	**Máquina**	**C 1**	**C 2**	**C 3**	**C 4**	**C 5**
Todos	41.2	58.8	7.7	13.2	19.2	21.4	38.5
Válidos	40.2	59.8	8.0	14.2	19.8	19.2	38.8

Cuadro 4.7: Diferencia de porcentaje de votos entre todos los jueces y los jueces válidos. Los jueces válidos son todos menos los jueces 11, 19, 20, 23 y 24.

4.3.3. Evaluación de los resultados

Luego del análisis de los resultados otorgados por las métricas basadas en heurísticas se puede ver que el modelo utilizado tiene mejores resultados que el algoritmo base, pero se mantiene por debajo del modelo implementado por los autores. Con el objetivo de entender la magnitud de las diferencias de estos valores, en el cuadro 4.8 se presenta una comparativa para cada métrica entre el resultado del modelo del autor (*SeqGan*), el modelo adversario que da resultados inferiores (*Al Gen Man*) y la línea base.

Modelo	***Average***	***Greedy***	***Extrema***
Al Gen (Man)	0.106	0.019	0.057
Baseline	0.226	0.058	0.235

Cuadro 4.8: Diferencia entre el peor modelo adversario y el *baseline* con el modelo *SeqGan* de Li et al. (2017).

Como se puede ver, la diferencia más significativa está en la métrica *Vector Extrema*. El modelo *Al Gen Man* muestra valores muy cercanos al modelo *SeqGan*, mientras que la línea base tiene dos puntos de diferencia con este. Por otro lado, teniendo en cuenta que la medida *Greedy Matching* presenta valores bajos para todos los modelos,

la línea base muestra una gran diferencia contra *SeqGan*. Finalmente, para *Embedding Average* se ve que el modelo *Al Gen Man* está un punto porcentual por debajo de *SeqGan*, mientras que la línea base está a 2.2. Estos resultados son de esperarse dado que el algoritmo base es simple y el implementado por los autores tiene más capacidad de procesamiento y de datos.

En general, en la evaluación humana, los jueces votaron de forma similar salvo por los diálogos que se presentan en el cuadro 4.4 y 4.5. Esta evaluación además muestra que si bien la mayoría de los jueces detectan que las oraciones son generadas por los modelos, a la hora de evaluar su calidad en base a la selección de un valor numérico dentro de un rango, tienden a seleccionar los valores tres, cuatro y cinco (calidad media o alta). Esto determina que en general las oraciones tienen cierto sentido, son naturales y permiten la continuación del diálogo.

Una de las observaciones sobre los formularios de la evaluación humana es que los jueces no cuentan con el contexto completo de los diálogos, lo cual puede influir en la elección de las respuestas. Además las clases dos, tres y cuatro no tienen una referencia asociada, lo que puede generar que jueces diferentes puedan asociar a estos valores un significado de calidad diferente.

Capítulo 5

Conclusión

El objetivo de este trabajo es utilizar técnicas de aprendizaje por refuerzo y redes generativas adversarias para construir un modelo capaz de generar diálogo escrito. Para llevar esta tarea a cabo se construyen dos modelos, uno por cada género. Estos están basados en la arquitectura GAN y son entrenados utilizando un corpus construido a partir de diálogos de películas, en los cuales los participantes son actores de diferente género.

En búsqueda de generar diálogos de diferentes géneros, se analizan guiones de películas románticas y se selecciona *Cornell Movie Dialogs Corpus*, que está formado por diálogos de películas de diferentes géneros en inglés. Para que pueda ser utilizado por el modelo es necesario realizar un procesamiento previo. Si bien la arquitectura utilizada es GAN, se analizaron otros modelos y arquitecturas con las cuales construir los modelos.

La evaluación de las respuestas generadas se realiza mediante métricas basadas en heurísticas y evaluación por jueces. La primeras de ellas consisten en métricas calculadas a partir de la representación vectorial de las oraciones generadas. Para esta evaluación se emplea como punto base de comparación un algoritmo que retorna una respuesta utilizando la similitud de las representaciones vectoriales de las oraciones. La evaluación por jueces consiste en dos experimentos: uno de ellos está inspirado en el test Turing y el otro busca determinar la calidad de la respuesta a través de la selección de un valor numérico dentro de un rango.

La evaluación basada en heurísticas muestra que, si bien el modelo es mejor al algoritmo implementado como línea base, no alcanza la calidad del modelo implementado por Li et al. (2017). Este resultado es esperado por el hecho de utilizar un corpus de menor tamaño y configurar el modelo con una menor cantidad de neuronas en las capas ocultas. La evaluación humana muestra que la mayoría de los jueces detectaron que las oraciones fueron generadas por los modelos. Sin embargo, a la hora de evaluar su calidad a través de la selección de un valor numérico dentro de un rango, los jueces tienden a seleccionar valores neutros o buenos. Esto demuestra que las oraciones en general tienen cierto sentido, son naturales y permiten la continuación del diálogo. Dados los resultados anteriores, se considera que el modelo de este trabajo tiene una calidad aceptable.

Un punto de este trabajo a analizar en el futuro, que por falta de tiempo quedó fuera del alcance, es si los modelos entrenados tienen particularidades en base a su género. Por ejemplo, estudiar si para una entrada determinada el modelo que representa el género femenino retorna una respuesta diferente al masculino. Otro factor que se podría estudiar es si las respuestas generadas por un género presentan expresiones o patrones particulares al responder a un determinado diálogo.

Por otro lado, se podría realizar la misma evaluación pero considerando un corpus con mayor cantidad de diálogos y en idioma español. El corpus que se utiliza para este trabajo tiene un tamaño menor comparado con los que utilizan los trabajos relacionados.

Para mejorar la calidad de la evaluación humana es necesario contar con una mayor cantidad y calidad de jueces. Con calidad de jueces se refiere a, por ejemplo, jueces que sean nativos en el idioma que se está evaluando. También se puede evaluar la participación de jueces con conocimientos en lingüística.

Apéndice A

Anexo I: Relación entre índices y diálogos

En el siguiente cuadro se muestra la relación entre los diálogos utilizados en la evaluación humana y su índice. Como forma de simplificar, en la exposición de resultados y gráficas se utiliza este índice en lugar del diálogo completo.

Índice	**Diálogo**
d1	Actor 1: *simple, it's fun.* Actor 2: *let me see if you like it. You can hear it for fun in front of fun.*
d2	Actor 1: *it's 0:00 by my watch. But it moves a bit fast.* Actor 2: *let's ask.*
d3	Actor 1: *that sounds ok. I'll need it from the 00th to the 00th.* Actor 2: *do you valid valid valid two valid?*
d4	Actor 1: *not really. They are just opera costumes. Do you like the embroidery?* Actor 2: *I don't know. They are cut machines in, but they are machines machines.*
d5	Actor 1: *what about the twill? it washes well and won't lose shape.* Actor 2: *what about cotton?*
d6	Actor 1: *mark. Where have you been? I've been calling you all morning.* Actor 2: *I'm going to have to cancel your master.*
d7	Actor 1: *okay, what time is my meeting in london?* Actor 2: *in 00. 00. would you mind come in on monday at 00:00 in good-bye?*
d8	Actor 1: *(the customer gets the aspirin and vitamins.) that'll be $ 00.00.* Actor 2: *here's a twenty.*
d9	Actor 1: *I think this spring is a good time for us to start looking.* Actor 2: *I agree with you only for a tour.*
d10	Actor 1: *I envy them the sun.* Actor 2: *I hate the oil, mum.*
d11	Actor 1: *why not just tell him I'm sick? I'll ring him myself this afternoon.* Actor 2: *alright. Have you been to see a doctor?*
d12	Actor 1: *000 yuan a night, plus vat, including air-conditioner and tv.* Actor 2: *that's fine. Can I take a message at that line?*
d13	Actor 1: *next time you see him, please tell him I'm thinking of him.* Actor 2: *I'll do that you make a message.*
d14	Actor 1: *did you notice the characters on the lanterns?* Actor 2: *no. But I don't think I can read any more characters. Would you recommend me some characters are saying that far.*
d15	Actor 1: *sure I am. I've been writing a blog for almost three years.* Actor 2: *oh, it seems that I did n't know the story of the blogs.*
d16	Actor 1: *really? that's really wonderful. I'm so happy for her.* Actor 2: *but nobody is with a ladder. I have to be with a teacher.*
d17	Actor 1: *let's see. How do you feel about playing bridge?* Actor 2: *it's ok with me, but I don't know the projected clear.*

d18	Actor 1: *did you like the chanel bag that I got?* Actor 2: *you must have boyfriend some boyfriend. You must keep a gift for a new boyfriend.*
d19	Actor 1: really? I'd rather have the chicken feet to be honest. Actor 2: ok then, let's go. I could have some fresh soup salad.
d20	Actor 1: *I want to make a phone call. Can you show me how?* Actor 2: *of course, easy. Pick you a minute pick of your receiver.*
d21	Actor 1: *thank you. Where should I send your final phone bill?* Actor 2: *side side.*
d22	Actor 1: *good morning. Could I speak to frans smith please?* Actor 2: *frans smith? I think you may have the wrong number.*
d23	Actor 1: *yes, please. That is very kind of you.* Actor 2: *shall I make more sandwiches?*
d24	Actor 1: *things are tough all over. You should never give up!* Actor 2: *but how can I prevent deficit?*
d25	Actor 1: *yeah, I've been out of town.* Actor 2: *so where have you been?*
d26	Actor 1: *do you have any questions?* Actor 2: *yes, I do. the company pays off at 00 thirty.*
d27	Actor 1: *yes, here you are.* Actor 2: *thank you. Do you have any direct card with you? or I will do.*
d28	Actor 1: *what if you had three more top-notch programmers to help you?* Actor 2: *that would definitely make a quick food. But we need to do that.*
d29	Actor 1: *I was just thinking about you the other day.* Actor 2: *you were?*
d30	Actor 1: *come home late one more time, and you'll find your stuff outside.* Actor 2: *you want to leave me know this has just bring me?*
d31	Actor 1: *well, I've lost my wallet and my id card.* Actor 2: *oh, that's fantastic!*
d32	Actor 1: *the train? will I be flying back then?* Actor 2: *yes, we don't. It's very important and you've got it.*
d33	Actor 1: *I want it to be special. And you know that I always like hot food.* Actor 2: *ok then, I will ask them to get back you. Anything else?*
d34	Actor 1: *hmm. Hey, wait. This has an apple on it. Is this an apple computer?* Actor 2: *yes, it's made mcintosh. mcintosh are rooms in your new card. mcintosh?*
d35	Actor 1: *yes, I have worked for four years with a canadian company.* Actor 2: *tell me what you know about years, please.*
d36	Actor 1: *oh, my. Why there are still so many things to learn after graduation?* Actor 2: *you know what that will be at touch?*
d37	Actor 1: *have you ever been tested for allergies?* Actor 2: *peaches make me break out, but I don't have any allergies then.*
d38	Actor 1: *how do you get along with people whose characters are different from your own?* Actor 2: *I try to adapt myself.*
d39	Actor 1: *perfect!* Actor 2: *yes, it is said great! I want your help with these clients.*
d40	Actor 1: *listen, karen, I need your help. I don't know anyone here yet.* Actor 2: *I'm glad to help you. What do you do?*
d41	Actor 1: *then go to bed and have a good sleep.* Actor 2: *ok. When did you start in your paper?*
d42	Actor 1: *I need to see your ticket. Do you have it?* Actor 2: *here's my ticket here is here.*

d43	Actor 1: *my permanent address is apt. 000,000 zhongshan road, beijing.* Actor 2: *where is your birthplace?*
d44	Actor 1: *I don't have any symptoms of high blood pressure.* Actor 2: *why has it be high pain?*
d45	Actor 1: *yeah, I think I'll get this one-piece.* Actor 2: *why don't you get a bikinI?*
d46	Actor 1: *he's not in at the moment. Can I take a message?* Actor 2: *sure.*
d47	Actor 1: *anything else?* Actor 2: *yogurt, please. You are not familiar with the company. Second, steven.*
d48	Actor 1: *let me see. How about 0:00 at the gate of the club?* Actor 2: *fine, see you then!*

Apéndice B

Anexo II: Relación entre índices y diálogos

En el siguiente cuadro se muestra la cantidad de votos que tuvo cada diálogo en cada uno de los experimentos realizados.

Diálogo	**Turing**		**Escala**					**Total por diálogo**
	Humano	**Máquina**	**C 1**	**C 2**	**C 3**	**C 4**	**C 5**	
d1	2	53	10	23	8	2	1	99
d2	28	26	7	7	6	14	9	97
d3	1	10	8	1	0	0	0	20
d4	1	12	4	2	2	0	1	22
d5	13	1	0	0	2	0	4	20
d6	7	8	3	1	5	1	1	26
d7	2	12	7	0	1	0	1	23
d8	6	12	1	0	0	3	3	25
d9	2	4	1	1	1	3	2	14
d10	1	13	4	5	4	0	1	28
d11	11	6	0	1	3	2	7	30
d12	3	5	1	6	0	1	2	18
d13	1	9	1	3	6	0	0	20
d14	7	4	3	4	1	1	0	20
d15	9	2	0	1	4	1	2	19
d16	2	9	5	2	0	0	1	19
d17	5	9	2	4	4	5	0	29
d18	1	5	8	1	0	1	1	17
d19	12	2	0	0	0	5	5	24
d20	1	10	1	1	1	0	0	14
d21	1	16	9	2	3	0	0	31
d22	15	0	0	0	0	1	5	21
d23	9	3	1	1	4	4	4	26
d24	8	7	0	1	1	4	0	21
d25	9	0	0	0	0	0	9	18
d26	3	9	0	4	2	1	1	20
d27	6	9	2	0	1	3	1	22
d28	1	8	4	1	1	3	0	18
d29	9	5	0	0	2	0	8	24
d30	0	8	2	4	2	0	0	16
d31	9	4	1	1	6	2	2	25
d32	0	11	7	2	1	0	0	21
d33	7	5	1	0	5	3	3	24
d34	0	9	5	2	0	0	1	17
d35	1	6	2	4	2	0	0	15

d36	1	6	5	3	1	1	0	17
d37	9	9	0	0	2	2	4	26
d38	15	3	0	0	1	1	7	27
d39	6	6	2	2	1	4	2	23
d40	8	5	0	1	0	2	4	20
d41	3	8	4	6	3	2	0	26
d42	3	4	1	1	5	3	1	18
d43	5	4	0	0	2	2	4	17
d44	3	10	4	3	5	1	0	26
d45	7	2	0	0	0	1	10	20
d46	4	2	0	0	0	3	6	15
d47	2	8	2	0	0	0	1	13
d48	7	0	0	0	0	0	4	11
Tot. por cate.	266	379	118	101	98	82	118	
Tot. por exp.		645					517	
Tot. de votos							1162	

Bibliografía

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*.

Banerjee, S. & Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, (pp. 65–72). Association for Computational Linguistics.

Buduma, N. & Locascio, N. (2017). *Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms* (1st ed.)., (pp. 195–197). O'Reilly Media, Inc.

Forgues, G., Pineau, J., Larchevêque, J.-M., & Tremblay, R. (2014). Bootstrapping dialog systems with word embeddings. In *NIPS, modern machine learning and natural language processing workshop*, volume 2.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27* (pp. 2672–2680). Curran Associates, Inc.

Graesser, L. & Keng, W. (2019). *Foundations of Deep Reinforcement Learning: Theory and Practice in Python*, chapter 2. Addison Wesley.

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

Li, J., Luong, T., & Jurafsky, D. (2015). A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, (pp. 1106–1115). Association for Computational Linguistics.

Li, J., Monroe, W., Ritter, A., Jurafsky, D., Galley, M., & Gao, J. (2016). Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, (pp. 1192–1202). Association for Computational Linguistics.

Li, J., Monroe, W., Shi, T., Jean, S., Ritter, A., & Jurafsky, D. (2017). Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, (pp. 2157–2169)., Copenhagen, Denmark. Association for Computational Linguistics.

Li, Z., Kiseleva, J., & de Rijke, M. (2018). Dialogue generation: From imitation learning to inverse reinforcement learning. *CoRR*, *abs/1812.03509*.

Li, Z., Kiseleva, J., & de Rijke, M. (2019). Dialogue generation: From imitation learning to inverse reinforcement learning. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, (pp. 6722–6729). AAAI Press.

Mitchell, J. & Lapata, M. (2010). Composition in distributional models of semantics. *Cogn. Sci.*, *34*(8), 1388–1429.

Mitchell, T. M. (1997). *Machine Learning* (1 ed.). McGraw-Hill.

Olah, C. (2015). Understanding LSTM Networks. Colah.Github.Io. https://colah.github.io/posts/2015-08-Understanding-LSTMs.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, (pp. 311–318). Association for Computational Linguistics.

Rus, V. & Lintean, M. (2012). A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, (pp. 157–162). Association for Computational Linguistics.

Serban, I. V., Sordoni, A., Bengio, Y., Courville, A. C., & Pineau, J. (2016). Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, (pp. 3776–3783). AAAI Press.

Sordoni, A., Galley, M., Auli, M., Brockett, C., Ji, Y., Mitchell, M., Nie, J.-Y., Gao, J., & Dolan, B. (2015). A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (pp. 196–205). Association for Computational Linguistics.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27* (pp. 3104–3112). Curran Associates, Inc.

Turing, A. M. (1950). Computing machinery and intelligence. volume 59 of *New Series*, (pp. 433–460). Oxford University Press on behalf of the Mind Association.

Wieting, J., Bansal, M., Gimpel, K., & Livescu, K. (2016). Towards universal paraphrastic sentence embeddings. In *4th International Conference on Learning Representations, ICLR 2016, , Conference Track Proceedings*.

Yu, L., Zhang, W., Wang, J., & Yu, Y. (2017). Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, (pp. 2852–2858). AAAI Press.